Two-way automata and regular languages of overlapping tiles

Anne Dicky, David Janin

Two-way automata and regular languages of overlapping tiles

Anne Dicky and David Janin

LaBRI UMR 5800, Université de Bordeaux, Bordeaux INP
351, cours de la Libération, F-33405 Talence, FRANCE
{dicky|janin}@labri.fr

Abstract. We consider classes of languages of overlapping tiles, i.e., subsets of the McAlister monoid: the class *REG* of languages definable by Kleene's regular expressions, the class *MSO* of languages definable by formulas of monadic second-order logic, and the class *REC* of languages definable by morphisms into finite monoids. By extending the semantics of finite-state two-way automata (possibly with pebbles) from languages of words to languages of tiles, we obtain a complete characterization of the classes *REG* and *MSO*. In particular, we show that adding pebbles strictly increases the expressive power of two-way automata recognizing languages of tiles, but the hierarchy induced by the number of allowed pebbles collapses to level one.

1 Introduction

Background

One-dimensional overlapping tiles already appear in the 70's in *inverse semigroup theory* [37]. As elements of particular quotients of free inverse monoids [40, 33], known as monoids of McAlister [34, 33], the tiles came to the forefront again in the late 90's in *mathematical physics*, associated with tilings of the Euclidian space [29–31, 1]. Although implicitly, overlapping tiles also appear in *theoretical computer science* in the studies of zigzag codes and the underlying zigzag covers of finite, infinite or bi-infinite words [2, 8, 35, 1]. Oddly enough, our interest in languages of positive tiles came from application perspectives in *computational music theory* [21]. In particular, tiles and continuous variants may be used to describe advanced synchronization mechanisms between musical patterns [4, 27]; an approach that leads to new programming features for music system design [28, 18].

In *software engineering*, overlapping tiles may be seen as the possible concrete values of string objects extended with history-preserving memory capacities. This point of view turns out to provide a simple presentation of many properties satisfied by one-dimensional overlapping tiles; it also conveys most of the intuition that underlies the work presented here.

Let us thus assume that we are software developers trying to enrich the class of string objects with some history-preserving capacity.

More precisely, for every string object s, let $s \cdot a$ denote the result of *adding* some character a to the right of the string s, and let $s \cdot \bar{a}$ denote the result of *removing* a from the right of s. With a standard string, $s \cdot a \cdot \bar{a} = s$, and thus $s \cdot a \cdot \bar{a} \cdot b = s \cdot b$ for any character b. A history-preserving mechanism is a way to prevent a character to appear in s if a different character has previously occurred in the same position. Thus our extended strings should satisfy the property

$$s \cdot a \cdot \bar{a} \cdot b = \begin{cases} s \cdot b \text{ if } a = b \\ \text{undefined otherwise} \end{cases}$$

as if adding and removing the character a to the right of the string s create some footprint of that character in such a way that no other character can ever be put later on that position.

One-dimensional overlapping tiles describe the effects of the possible sequences of actions (additions or removals of characters) on these extended string objects; and thus the possible values of the objects themselves (as the effects of sequences of actions on the empty string). For instance, the effect of the sequence $\bar{a}abcba\bar{a}\bar{b}$ is described by the tile

$$a\ bc\ ba$$

where bc is the string to be added, while the left part a and the right part ba of the tile model the footprints left by the other actions. The composition of actions yields a monoid structure of tiles that turns out to be the inverse monoid of McAlister [37, 34].

These examples show that the model of one-dimensional overlapping tiles is a versatile model that can be used in many fields. However, the associated language theory can still be further developed. Indeed, the classical tools of formal language theory somehow fail to apply to inverse monoids [36, 46]. To be more precise, the expressive power induced by the usual formal language theoretic tools, namely, the automata induced by morphisms into finite monoids, collapses when applied to inverse monoids.

In this paper, we aim at developing a computer science-flavored formal language theory for overlapping tiles. Since adding or removing characters of extended string objects can be interpreted as movements of the reading head of a two-way automaton [42] on a classical string (adding a character corresponds to reading it from left to right, whereas removing it corresponds to reading it from right to left), finite-state two-way automata appear as natural and expressive candidates to define and study classes of tile languages.

Incidentally, following a habit quite developed in formal language theory, strings will be called words and characters will be called letters throughout the remainder of the text.

Outline

The monoid of one-dimensional overlapping tiles is presented in Section 2 (Theorem 2.8). A special emphasis is put on the way non-zero tiles are generated from linear walks (Theorem 2.31), thus rephrasing, in the context of one-dimensional tiles, the notion of free inverse monoid captured by the Wagner congruence (Lemma 2.28). The link with Pécuchet's notion of bisection [39] is specified at the end of the section (Remark 2.32).

To a specialist of inverse semigroups, most of the material presented in this section is quite straightforward. In particular, our presentation of the monoid of McAlister could be significantly simplified by defining it as a Rees' quotient of the free inverse monoid, following the classical Scheiblich-Munn presentation of free inverse monoids [44, 38]. We prefer a direct, standalone presentation to address a more general public, providing an alternative to Lawson's presentation [34] (see also [33], chap. 9, for a relationship with various other classes of semigroups).

From Section 3 we study languages of tiles, starting with the class *REG* of languages definable by regular expressions.

The class *REC* of languages of tiles recognizable by finite monoids is shown to be strictly included in the class *REG* (Theorem 3.11 and Corollary 3.14). Still, a nontrivial example of a language in *REC* is given (Example 3.16), illustrating the complete characterization of the class *REC* given in [26]. Further details on languages recognized by inverse monoids can be found in [36, 46].

Then we define the class *2WA* of languages accepted by two-way tile automata. Quite closely related with Pécuchet's study [39], two-way tile automata are standard two-way automata over

words with a semantics expressed in terms of tiles. Tiles are simply seen as the results of partial runs: runs that may start and stop anywhere on the input words. We show that the class *REG* of regular languages of tiles (definable by Kleene expressions) corresponds to the class *2WA* of languages of tiles recognized by finite-state two-way tile automata (Theorem 3.21).

Beyond regular languages, the class *MSO* of languages definable in Monadic Second-Order Logic is studied in Section 4. This class is shown to be both robust (Theorem 4.3) and simple (Theorem 4.7). It is strictly larger than *REG* (see Corollary 4.4).

As a consequence of robustness, it is shown that *MSO* contains the class k-*REG* of k-regular languages, i.e., tile languages that can be defined by Kleene expressions extended with the idempotent projection operator, with a nesting depth at most k. As a consequence of simplicity, it is shown (Theorem 4.12) that the class *MSO* equals the class *1-REG* (extended regular expressions with no nested projection operators).

We also prove that *MSO* is the class of languages recognized by finite-state two-way many-pebble automata. A simple correspondence between k-regular languages (k-*REG*) and k-pebble automata (k-*P2WA*) is shown to hold (Theorem 4.18). Thus one-pebble automata capture the whole class of many-pebble automata (Theorems 4.12 and 4.18). Indeed, the equality *1-P2WA* = *MSO* was first proved in Theorem 3.3 of [10], where a tile language is called a *trip*, and an MSO-definable tile language is called a *regular trip*.

Shepherdson's theorem and analogous known results for pebble automata are obtained as immediate corollaries (Corollaries 4.9 and 4.21).

To summarize, we prove that for every integer $k \geq 1$:

$$REC \subsetneq REG = 2WA \subsetneq 1\text{-}REG = 1\text{-}P2WA = k\text{-}REG = k\text{-}P2WA = MSO$$

All these results support the long-standing intuition [39, 5, 32] that the theory of inverse monoids *is* a powerful tool in the study of two-way automata. Indeed, all proofs presented here are quite simple.

Although the expressiveness hierarchy induced by k-regular expressions (resp. k-pebble automata) collapses to its first level, the notion of k-regular expressions (resp. k-pebble automata) is still worth being studied since, following [17], it is conjectured that k-regular expressions induces a succinctness hierarchy (Conjecture 4.13).

Related works

Two-way automata have been the subject of many studies. This can be explained by their intriguing combinatorial complexity.

For instance, Rabin-Scott-Shepherdson's result [42, 45] that two-way automata are as expressive on words as one-way automata was long considered difficult [48]. More precisely, the capacity of two-way automata to read each letter an unbounded number of times makes the structure of two-way automata runs difficult to analyze. This is particularly clear in Pécuchet and Birget's algebraic studies of two-way automata [39, 5], in which two-way runs give rise to a rich combinatorial structure. A similar complexity is illustrated by Globerman and Harel's result [17] that the number of allowed pebbles in two-way automata induces a "succinctness" hierarchy: each additional pebble provides inherent exponential power.

Still, gaining a full understanding of two-way automata, with or without pebbles, remains a challenging topic (see, e.g., [16]). The classical theory of (one-way) finite-state automata has benefited from a rich algebraic language theory that led, and still leads, to many decision algorithms [41]. But, as already observed by Birget [5], there is no similar algebraic characterization of two-way automata that does not amount to reducing two-way automata to one-way automata.

Further studies on languages of overlapping tiles [20, 24], infinite tiles [7], or on languages of birooted trees [26, 22, 25], show that some progress can be done along Birget's long-standing open question [5].

2 Overlapping tiles

Here we give a description of monoids of one-dimensional overlapping tiles, and show that they are isomorphic to monoids of McAlister [34]. The link between tiles and two-way linear walks on words is formalized by an onto morphism from walks to tiles whose kernel is indeed the Wagner congruence.

2.1 Preliminaries

Given a finite alphabet A, let A^* be the free monoid generated by A and let 1 be the neutral element. The concatenation of two words u and v is denoted by uv. The length of a word u is denoted by $|u|$.

Let $\leq_p$ stand for the (partial) prefix order over A^*, let $\leq_s$ stand for the suffix order, and let $\vee_p$ (resp. $\vee_s$) denote the join operator for the prefix (resp. suffix) order. For all words u and v, the word $u \vee_p v$ (resp. $u \vee_s v$) is the least word of which both u and v are prefixes (resp. suffixes) when it exists. The extended monoid $A^* \cup \{0\}$ with $00 = 0$ and $0u = u0 = 0$ for every word u, with the prefix order extended to 0 by $0 \leq_p 0$ and $u \leq_p 0$ for every word u, is a lattice; in particular, $u \vee_p v = 0$ whenever neither u is a prefix of v, nor v is a prefix of u. Symmetric properties hold in the suffix lattice.

Given a disjoint copy $\bar{A} = \{\bar{a} : a \in A\}$ of the alphabet A, let $u \mapsto \overline{u}$ be the mapping from $(A \cup \bar{A})^*$ to itself inductively defined by $\overline{1} = 1$, $\overline{au} = \overline{u}\,\bar{a}$ and $\overline{\bar{a}u} = \overline{u}\,a$ for every letter $a \in A$ and every word $u \in (A \cup \bar{A})^*$. The mapping $u \mapsto \overline{u}$ is involutive, i.e., for every word $u \in (A \cup \bar{A})^*$ we have $\overline{\overline{u}} = u$. It is also an antimorphism of the free monoid $(A \cup \bar{A})^*$, i.e., for all words u and $v \in (A \cup \bar{A})^*$ we have $\overline{uv} = \overline{v}\,\overline{u}$. For all $u \in (A \cup \bar{A})^*$, the word $\overline{u}$ is called the *syntactic inverse* of u.

The free group $FG(A)$ generated by A is the quotient of $(A \cup \bar{A})^*$ by the least congruence $\simeq$ such that, for every letter $a \in A$, $a\bar{a} \simeq 1$ and $\bar{a}a \simeq 1$. Let $\to$ denote the rewriting relation induced by the rules $a\bar{a} \to 1$ and $\bar{a}a \to 1$ for every $a \in A$. It is well known that every congruence class $[u] \in FG(A)$ contains a unique element $red(u)$ (the *reduced form* of u) irreducible with respect to $\to$, i.e., containing no subword of either form $a\,\bar{a}$ and $\bar{a}\,a$. It follows that the elements of $FG(A)$ may be defined as words of the form $red(u)$ with $u \in (A \cup \bar{A})^*$, with the group product defined by $u \cdot v = red(uv)$ for all $u, v \in FG(A)$. Since we have $u \cdot \bar{u} = 1 = \bar{u} \cdot u$, in the free group $FG(A)$, the syntactic inverses are the group inverses. By extension, for all words u and v in $(A \cup \bar{A})^*$, let also $u \cdot v$ denote the reduced form of uv. The following are well-known properties of reduction: for all $u, v, w \in (A \cup \bar{A})^*$,

- ▷ if $u, v \in A^*$ or $u, v \in \bar{A}^*$ then $u \cdot v = uv$,
- ▷ $\overline{u \cdot v} = \bar{v} \cdot \bar{u}$,
- ▷ $(u \cdot v) \cdot w = u \cdot (v \cdot w)$.

We will also use the properties stated in the following lemma.

Lemma 2.1. *Let $u, v \in A^*$ and $w \in (A \cup \bar{A})^*$. The following properties hold:*

1. *$u \vee_p v \neq 0$ if and only if $\bar{u} \cdot v \in A^* \cup \bar{A}^*$, and $u \vee_s v \neq 0$ if and only if $u \cdot \bar{v} \in A^* \cup \bar{A}^*$,*
2. *if $u \vee_p v \neq 0$ and $w \cdot u \in A^*$ then $w \cdot (u \vee_p v) \in A^*$, and if $u \vee_s v \neq 0$ and $u \cdot w \in A^*$ then $(u \vee_s v) \cdot w \in A^*$,*

3. *if* $w \in A^*$ *then* $w \cdot (u \vee_p v) = w \cdot u \vee_p w \cdot v$ *and* $(u \vee_s v) \cdot w = u \cdot w \vee_s v \cdot w$,
4. *if* $w \in A^*$ *then* $u \vee_p v = \bar{w} \cdot (w \cdot u \vee_p w \cdot v)$ *and* $u \vee_s v = (u \cdot w \vee_s v \cdot w) \cdot \bar{w}$.

Proof.

1. u is a prefix of v if and only if $\bar{u} \cdot v \in A^*$; v is a prefix of u if and only if $\bar{v} \cdot u \in A^*$, or equivalently $\bar{u} \cdot v = \overline{\bar{v} \cdot u} \in \bar{A}^*$. The proof for $\vee_s$ is symmetrical.
2. The property is trivial if $u \vee_p v = u$. Otherwise, $v = uv'$ for some $v' \in A^*$. Then $w \cdot (u \vee_p v) = w \cdot (u \cdot v') = (w \cdot u) \cdot v' \in A^*$. The proof for $\vee_s$ is symmetrical.
3. Well known properties of prefix, suffix and concatenation in A^*.
4. Immediate consequence of the previous property.

□

For a monoid M and $x, y \in M$, we say that y is an *inverse* of x if $xyx = x$ and $yxy = y$. A monoid M is an *inverse monoid* if every element of M has a unique inverse in M. The following lemma is well-known (see [40, 33]).

Lemma 2.2. *Let M be a monoid such that (i) every element of M has an inverse, and (ii) all idempotent elements commute (i.e., if $xx = x$ and $yy = y$ then $xy = yx$). Then M is an inverse monoid.*

Proof. Let x be an element of M. For any inverse y of x, $(xyx)y = xy$ and $y(xyx) = yx$, thus xy and yx are idempotent.

Let y_1 and y_2 be inverses of x. Since xy_1 and xy_2 are idempotent, they commute, and we have

$$xy_2 = (xy_1x)y_2 = (xy_1)(xy_2) = (xy_2)(xy_1) = (xy_2x)y_1 = xy_1$$

thus $xy_2 = xy_1$. Symmetrically it can be shown that $y_2x = y_1x$. It follows that $y_2 = y_2xy_2 = y_2xy_1 = y_1xy_1 = y_1$, and thus the inverse of x is unique. □

Conversely, it can be shown that the idempotents of an inverse monoid commute.

By Wagner's theorem, the *free inverse monoid FIM*(A) generated by A can be defined (see [33]) as the quotient of $(A \cup \bar{A})^*$ by the Wagner congruence $\simeq_W$, i.e., the least congruence such that $u\bar{u}u \simeq_W u$ and $u\bar{u}v\bar{v} \simeq_W v\bar{v}u\bar{u}$ for all $u, v \in (A \cup \bar{A})^*$. We refer the interested reader to the books [40, 33] for details of the inverse semigroup theory.

2.2 The inverse monoid of tiles

We define here the notion of tile and the related notion of product. The resulting structure is shown to be an inverse monoid.

Definition 2.3 (Tile). A *tile* over the alphabet A is a triple of words $u = (u_1, u_2, u_3) \in A^* \times (A^* \cup \bar{A}^*) \times A^*$ such that $u_1 \cdot u_2$ and $u_2 \cdot u_3$ are both in A^*; i.e., if $u_2 \in \bar{A}^*$, then its syntactic inverse $\overline{u_2}$ is a suffix of u_1 and a prefix of u_3. The word u_2 is called the *root path* of the tile. When $u_2 \in A^*$ we say that u is a *positive tile*. When $u_2 \in \bar{A}^*$ we say that u is a *negative tile*. Observe that a negative tile is of the form $(v_1v_2, \overline{v_2}, v_2v_3)$ with $v_1, v_2, v_3 \in A^*$.

A positive tile $u = (u_1, u_2, u_3)$ is conveniently drawn as a (linear, unidirectional and left-to-right) Munn's birooted word tree [38] as depicted in Figure 1 where the dangling input arrow, marking the beginning of the root path and called the *input root* of the tile, appears to the left of the dangling output arrow, marking the end of the root path and called the *output root* of the tile.

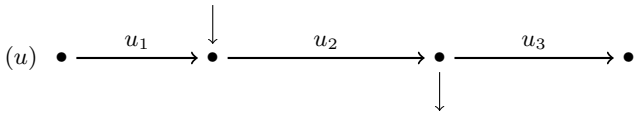


Fig. 1. A positive tile (u_1, u_2, u_3).

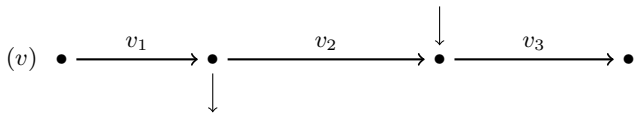


Fig. 2. A negative tile $(v_1v_2, \overline{v}_2, v_2v_3)$.

A negative tile of the form $v = (v_1v_2, \overline{v}_2, v_2v_3) \in A^* \times \bar{A}^* \times A^*$ is also drawn as a birooted word tree as depicted in Figure 2 where the input root now appears to the right of the output root.

The *domain* of a tile $u = (u_1, u_2, u_3)$ is $u_1 \cdot u_2 \cdot u_3$, the reduced form of $u_1u_2u_3$ (if u is a negative tile $(v_1v_2, \overline{v}_2, v_2v_3)$, its domain is $v_1v_2v_3$).

The set of positive tiles (resp. negative tiles) is denoted by $T^+(A)$ (resp. $T^-(A)$). The set of tiles is denoted by $T(A)$. Let $T_0(A) = T(A) \cup \{0\}$ denote the set of tiles extended with a *zero tile* 0.

Definition 2.4 (Product of tiles).
The *product* of two non-zero tiles $u = (u_1, u_2, u_3), v = (v_1, v_2, v_3) \in T(A)$ is defined as

$$u \cdot v = (((u_1 \cdot u_2) \vee_s v_1) \cdot \overline{u}_2, u_2 \cdot v_2, \overline{v}_2 \cdot (u_3 \vee_p (v_2 \cdot v_3)))$$

when both pattern-matching conditions $(u_1 \cdot u_2) \vee_s v_1 \neq 0$ and $u_3 \vee_p (v_2 \cdot v_3) \neq 0$ are satisfied, and 0 otherwise. The product is extended to the zero tile by $u \cdot 0 = 0 \cdot u = 0$ for every $u \in T_0(A)$.

Remark 2.5. When A is a single-letter alphabet, the product of two non-zero tiles is always a non-zero tile. The set $T(A)$ of non-zero tiles is known to be the free inverse monoid of one generator. Unless explicitly stated, we assume in this paper that A contains at least two distinct letters.

Example 2.6. Let a, b, c and d be distinct letters of A. Then $(a, b, c) \cdot (b, c, d) = (a, bc, d)$, but $(a, b, c) \cdot (a, c, d) = 0$ (the left-matching constraint is violated since neither ab is a suffix of a, nor a is a suffix of ab).

Remark 2.7. The product $u \cdot v$ of two matching positive tiles $u = (u_1, u_2, u_3)$ and$v = (v_1, v_2, v_3)$ is depicted in Figure 3. The product $u \cdot v$ of a positive tile $u = (u_1, u_2, u_3)$ and a matching negative tile $v = (v_1v_2, \overline{v}_2, v_2v_3)$ is depicted in Figure 4.

These figures show that, in terms of birooted words, the product $u \cdot v$ is obtained intuitively by identifying the output root of u with the input root of v. Taking into account this identification, the domain of $u \cdot v$ is the smallest word that contains the domains of u and v (which exists if only if u and v match), the input root of $u \cdot v$ is the input root of u, and the output root of $u \cdot v$ is the output root of v. These are key features to ensure that the product is associative.

Theorem 2.8. *The set $T_0(A)$ of tiles over the alphabet A, equipped with the product of tiles, is a monoid with neutral element $1 = (1, 1, 1)$.*

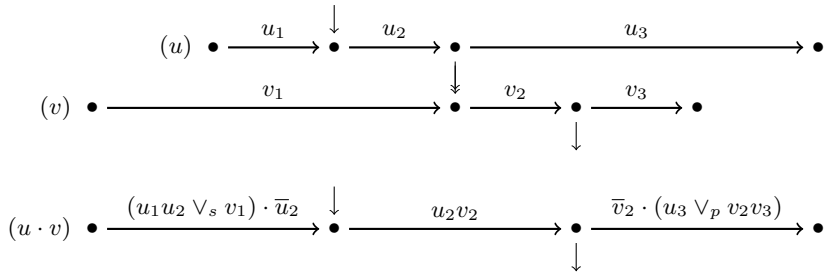


Fig. 3. The product of two positive tiles.

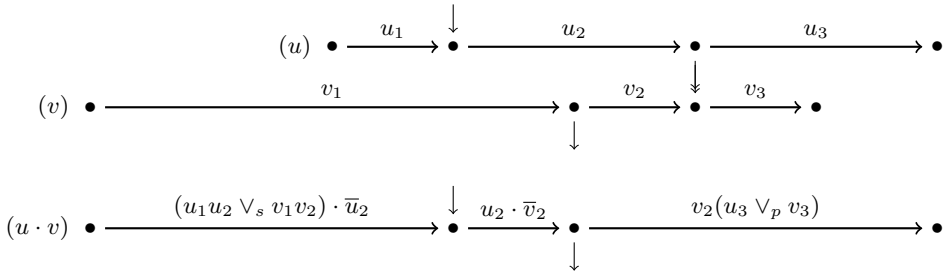


Fig. 4. The product of a positive tile and a negative tile.

Proof. We have to prove that the product of tiles is a sound (well-defined) associative operation.

Soundness. Let $u = (u_1, u_2, u_3)$ and $v = (v_1, v_2, v_3)$ be tiles such that $u \cdot v \neq 0$. Since $u_1 \cdot u_2$, v_1 and $(u_1 \cdot u_2) \cdot \bar{u}_2 = u_1$ are in A^*, item 2 of Lemma 2.1 implies that the first component $((u_1 \cdot u_2) \vee_s v_1) \cdot \overline{u}_2$ of the product tile $u \cdot v$ is in A^*. Similarly, its third component $\overline{v}_2 \cdot (u_3 \vee_p (v_2 \cdot v_3))$ is also in A^*.

We now prove that the second component $u_2 \cdot v_2$ is in $A^* \cup \bar{A}^*$. It is obvious if the tiles u and v are both positive or both negative. Suppose $u_2 \in A^*$ and $v_2 \in \bar{A}^*$. Since $u_2 \leq_s u_1 u_2$ and $\overline{v}_2 \leq_s v_1$, we have $u_2 \vee_s \overline{v}_2 \leq_s u_1 u_2 \vee_s v_1 \neq 0$, hence $u_2 \vee_s \overline{v}_2 \neq 0$ and thus $u_2 \cdot v_2 \in A^* \cup \bar{A}^*$ by item 1 of Lemma 2.1. The proof for $u_2 \in \bar{A}^*$ and $v_2 \in A^*$ is similar, using $\overline{u}_2 \leq_p u_3$ and $u_3 \vee_p v_2 v_3 \neq 0$.

Finally we prove that $u \cdot v$ is a tile. The reduced concatenation of its first two components, $((u_1 \cdot u_2) \vee_s v_1) \cdot \overline{u}_2 \cdot u_2 \cdot v_2 = (v_1 \vee_s (u_1 \cdot u_2)) \cdot v_2$, is in A^* by item 2 of Lemma 2.1 since $v_1 \cdot v_2 \in A^*$. Similarly, since $u_2 \cdot u_3 \in A^*$, the reduced concatenation of the last two components $u_2 \cdot v_2 \cdot \overline{v}_2 \cdot (u_3 \vee_p (v_2 \cdot v_3)) = u_2 \cdot (u_3 \vee_p (v_2 \cdot v_3))$ is also in A^*.

Associativity. Let $u = (u_1, u_2, u_3)$, $v = (v_1, v_2, v_3)$ and $w = (w_1, w_2, w_3)$ be non-zero tiles. Then $(u \cdot v) \cdot w = (x_1, x_2, x_3)$ with

$$\begin{aligned} x_1 &= ((u_1 \cdot u_2 \vee_s v_1) \cdot v_2 \vee_s w_1) \cdot \overline{v}_2 \cdot \overline{u}_2 \\ x_2 &= u_2 \cdot v_2 \cdot w_2 \\ x_3 &= \overline{w}_2 \cdot (\overline{v}_2 \cdot (u_3 \vee_p v_2 \cdot v_3) \vee_p w_2 \cdot w_3) \end{aligned}$$

provided that the corresponding pattern-matching conditions are satisfied.
Similarly, $u \cdot (v \cdot w) = (y_1, y_2, y_3)$ with

$$\begin{aligned} y_1 &= (u_1 \cdot u_2 \vee_s (v_1 \cdot v_2 \vee_s w_1) \cdot \overline{v}_2) \cdot \overline{u}_2 \\ y_2 &= u_2 \cdot v_2 \cdot w_2 = x_2 \\ y_3 &= \overline{w}_2 \cdot \overline{v}_2 \cdot (u_3 \vee_p v_2 \cdot (v_3 \vee_p w_2 \cdot w_3)) \end{aligned}$$

again provided that the pattern-matching expressions are non-zero.

To prove that $(u \cdot v) \cdot w = u \cdot (v \cdot w)$ we distinguish two cases.

First assume that $v_2 \in A^*$. From item 3 of Lemma 2.1, it follows that

$$x_1 = (u_1 \cdot u_2 \cdot v_2 \vee_s v_1 \cdot v_2 \vee_s w_1) \cdot \overline{v}_2 \cdot \overline{u}_2$$

with the equivalent pattern-matching condition $u_1 \cdot u_2 \cdot v_2 \vee_s v_1 \cdot v_2 \vee_s w_1 \neq 0$.
From item 2 of Lemma 2.1, when $v_1 \cdot v_2 \vee_s w_1 \neq 0$ we have $(v_1 \cdot v_2 \vee_s w_1) \cdot \overline{v}_2 \in A^*$. Applying item 4 of Lemma 2.1 to $u := u_1 \cdot u_2$, $v := (v_1 \cdot v_2 \vee_s w_1) \cdot \overline{v}_2$ and $w := v_2$, we obtain $u_1 \cdot u_2 \vee_s (v_1 \cdot v_2 \vee_s w_1) \cdot \overline{v}_2 = (u_1 \cdot u_2 \cdot v_2 \vee_s (v_1 \cdot v_2 \vee_s w_1) \cdot \overline{v}_2 \cdot v_2) \cdot \overline{v}_2 = (u_1 \cdot u_2 \cdot v_2 \vee_s v_1 \cdot v_2 \vee_s w_1) \cdot \overline{v}_2$ and thus $y_1 = x_1$. Similarly, it can be shown that $x_3 = y_3 = \overline{w}_2 \cdot \overline{v}_2 \cdot (u_3 \vee_p v_2 \cdot v_3 \vee_p v_2 \cdot w_2 \cdot w_3)$.

Assume now that $v_2 \in \bar{A}^*$. From item 3 of Lemma 2.1 with $w := \overline{v}_2$, we obtain

$$x_1 = y_1 = (u_1 \cdot u_2 \vee_s v_1 \vee_s w_1 \cdot \overline{v}_2) \cdot \overline{u}_2$$
$$x_3 = y_3 = \overline{w}_2 \cdot (\overline{v}_2 \cdot u_3 \vee_p v_3 \vee_p w_2 \cdot w_3)$$

which conclude the proof of associativity.
Neutral element. We conclude by observing that clearly $(1, 1, 1)$ is a neutral element. □

Observe that the set $T_0^+(A)$ of positive tiles (resp. the set $T_0^-(A)$ of negative tiles) extended with 0 is a submonoid of $T_0(A)$.

Obviously, a non-zero tile $u = (u_1, u_2, u_3)$ is *idempotent* for the product, that is, it satisfies $u \cdot u = u$, if and only if $u_2 = 1$, i.e., if and only if u is both positive and negative. Let $E(A)$ denote the set of non-zero idempotent tiles and let $E_0(A) = E(A) \cup \{0\}$ denote the set of all idempotent tiles. We have $E(A) = T^+(A) \cap T^-(A)$ and $E_0(A) = T_0^+(A) \cap T_0^-(A)$.

Lemma 2.9. *The set $E_0(A)$ of all idempotent tiles is a commutative submonoid of $T_0(A)$.*

Proof. Let $u = (u_1, 1, u_3)$ and $v = (v_1, 1, v_3)$ be non-zero idempotent tiles. Then $u \cdot v \neq 0$ if and only if $u_1 \vee_s v_1 \neq 0$ and $u_3 \vee_p v_3 \neq 0$ (thus if and only if $v \cdot u \neq 0$), in which case $u \cdot v = (u_1 \vee_s v_1, 1, u_3 \vee_p v_3) = v \cdot u$. □

Definition 2.10 (Inverse). The *inverse* u^{-1} of a non-zero tile $u = (u_1, u_2, u_3)$ is the tile obtained by swapping its input root and its output root: thus

$$u^{-1} = (u_1 \cdot u_2, \overline{u}_2, u_2 \cdot u_3)$$

This definition is extended to zero by letting $0^{-1} = 0$.

The notion of inverse is depicted in Figure 5.

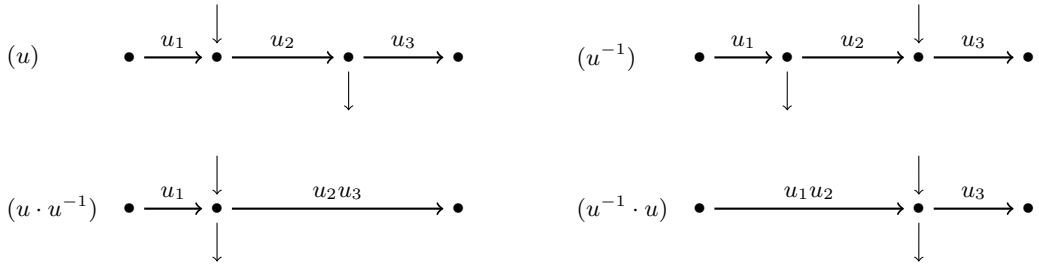


Fig. 5. Inverses and associated idempotents.

The following properties are straightforward.

Lemma 2.11. *For all tiles $u, v \in T_0(A)$:*

- ▷ $(u^{-1})^{-1} = u$
- ▷ $(u \cdot v)^{-1} = v^{-1} \cdot u^{-1}$
- ▷ $u^{-1} = u$ *if and only if* u *is idempotent*
- ▷ $u \cdot u^{-1}$ *and* $u^{-1} \cdot u$ *are idempotents.*

Note that the last item of this lemma is an immediate consequence of the first three items.

The next theorem shows that the monoid $T_0(A)$, equipped with the inverse operation, is indeed an inverse monoid.

Theorem 2.12. *The monoid of tiles $T_0(A)$ is an inverse monoid.*

Proof. It is straightforward to check that if u is a non-zero tile (u_1, u_2, u_3), then $u \cdot u^{-1} = (u_1, 1, u_2 \cdot u_3)$ and $u^{-1} \cdot u = (u_1 \cdot u_2, 1, u_3)$ (see Figure 5). It follows that $u \cdot u^{-1} \cdot u = u$ and $u^{-1} \cdot u \cdot u^{-1} = u^{-1}$. Thus every element of $T_0(A)$ has an inverse. By Lemma 2.11, the idempotents of $T_0(A)$ commute. By Lemma 2.2, $T_0(A)$ is an inverse monoid. □

2.3 Words and tiles

We define here the notion of canonical morphism from $(A \cup \bar{A})^*$ into $T_0(A)$. This leads to the study of linear walks and to the proof that the monoid of tiles is the monoid of McAlister [34].

Definition 2.13 (Canonical morphism). Let $\theta : (A \cup \bar{A})^* \to T_0(A)$ be the monoid morphism defined by

$$\theta(a) = (1, a, 1) \quad \text{and} \quad \theta(\bar{a}) = (a, \bar{a}, a)$$

for every $a \in A$.

The next lemmas, given with standalone proofs, are well-known consequences of the fact that $T_0(A)$ is an inverse monoid (Theorem 2.12) and that $\theta(\overline{a}) = \theta(a)^{-1}$ for every $a \in A$.

Lemma 2.14. *For every $w \in (A \cup \bar{A})^*$, we have $\theta(\overline{w}) = \theta(w)^{-1}$.*

Proof. By induction on the length of w, since for every $a \in A \cup \bar{A}$ we have $\theta(\overline{a}) = \theta(a)^{-1}$, and if $\theta(\overline{w}) = \theta(w)^{-1}$ then $\theta(\overline{aw}) = \theta(\overline{w}) \cdot \theta(\overline{a}) = \theta(w)^{-1} \cdot \theta(a)^{-1}$ hence $\theta(\overline{aw}) = \theta(aw)^{-1}$. □

Lemma 2.15. *The canonical morphism preserves the Wagner congruence, i.e., for every $w_1, w_2 \in (A \cup \bar{A})^*$, if $w_1 \simeq_W w_2$ then $\theta(w_1) = \theta(w_2)$.*

Proof. This amounts to proving that $\theta(u\bar{u}u) = \theta(u)$ and $\theta(u\bar{u}v\bar{v}) = \theta(v\bar{v}u\bar{u})$ for all $u, v \in (A \cup \bar{A})^*$.

Let $u \in (A \cup \bar{A})^*$. We have $\theta(u\bar{u}u) = \theta(u) \cdot \theta(\bar{u}) \cdot \theta(u)$. By Lemma 2.14 we have $\theta(\bar{u}) = \theta(u)^{-1}$, hence $\theta(u\bar{u}u) = \theta(u)$ since $T_0(A)$ is an inverse monoid.

Let $u, v \in (A \cup \bar{A})^*$. By the same argument, we have $\theta(u\bar{u}) = \theta(u) \cdot \theta(u)^{-1}$ and $\theta(v\bar{v}) = \theta(v) \cdot \theta(v)^{-1}$, hence both $\theta(u\bar{u})$ and $\theta(v\bar{v})$ are idempotents by Lemma 2.11, and thus commute. □

Remark 2.16. For every $w \in A^*$, we have $\theta(w) = (1, w, 1)$ (by induction on the length of w) and $\theta(\bar{w}) = (1, w, 1)^{-1} = (w, \overline{w}, w)$ (by Lemma 2.14). Thus $\theta(A^*)$ is a submonoid of $T_0(A)$ isomorphic to A^*, and the monoid of tiles $T_0(A)$ can be seen as an extension of the free monoid A^*. Similarly $T_0(A)$ can also be seen as an extension of the free monoid $\bar{A}^*$.

The following notion of left and right projections, central in inverse semigroup theory since they characterize Green's left and right classes (see [33], Chap. 3.2), yields a simple proof that θ is surjective.

Definition 2.17 (Left and right projections). For every $u \in T_0(A)$, let $u^L = u^{-1} \cdot u$ be the *left projection* of the tile u, and $u^R = u \cdot u^{-1}$ be its *right projection*.

As observed in the proof of Theorem 2.12, if $u = (u_1, u_2, u_3)$ is a non-zero tile, then we have $u^L = (u_1 \cdot u_2, 1, u_3)$ and $u^R = (u_1, 1, u_2 \cdot u_3)$. Examples of left and right projections have already been depicted in Figure 5. The next lemma shows how tiles are related to words via left and right projections.

Lemma 2.18. *For every positive tile $u = (u_1, u_2, u_3)$ we have $u = \theta(u_1)^L \cdot \theta(u_2) \cdot \theta(u_3)^R$ and for every negative tile $v = (u_1u_2, \overline{u}_2, u_2u_3)$ we have $v = \theta(u_3)^R \cdot \theta(\overline{u}_2) \cdot \theta(u_1)^L$.*

Proof. By the first sentence of Remark 2.16, for every word $w \in A^*$ we have $\theta(w)^L = (w, 1, 1)$ and $\theta(w)^R = (1, 1, w)$. It is easy to check that if $u = (u_1, u_2, u_3)$ is a positive tile, then $(u_1, 1, 1) \cdot (1, u_2, 1) \cdot (1, 1, u_3) = (u_1, u_2, u_3)$, hence $u = \theta(u_1)^L \cdot \theta(u_2) \cdot \theta(u_3)^R$ and, taking the inverses, $u^{-1} = \theta(u_3)^R \cdot \theta(\overline{u}_2) \cdot \theta(u_1)^L$. □

It follows that:

Lemma 2.19. *The canonical morphism $\theta : (A \cup \bar{A})^* \to T_0(A)$ is surjective.*

Proof. By Lemma 2.14, $\theta(w)^L = \theta(\bar{w}w)$ and $\theta(w)^R = \theta(w\bar{w})$ for every $w \in (A \cup \bar{A})^*$. Hence, by Lemma 2.18, every non-zero tile is a finite product of elements of $\theta((A \cup \bar{A})^*)$, hence an element of $\theta((A \cup \bar{A})^*)$. Since 0 may be obtained as $\theta(a\bar{b})$ where a and b are distinct letters of A, we conclude that the morphism θ is surjective. □

As an immediate consequence:

Lemma 2.20. *The monoid $T_0(A)$ is finitely generated from $\theta(A \cup \bar{A})$.*

Observe that the submonoid $T_0^+(A)$ of positive tiles is *not* finitely generated by products. However, we have:

Lemma 2.21. *The monoid $T_0^+(A)$ of positive tiles is finitely generated from $\theta(A)$ by products and left and right projections.*

Proof. Immediate from Lemma 2.18 observing that $0 = \theta(a) \cdot \theta(b)^L$ for $a \neq b$. □

Remark 2.22. Projections are a corner stone of the notion of quasi-recognizability [20, 19] or the notion of non-deterministic tile automata [24] and their closure properties [9], and they play a major role in the language theory of infinite tiles [7]. In recent developments of inverse semigroup theory, the projections are used to define "almost inverse" for more general classes of semigroups such as *ample monoids* (see, e.g., [14]).

2.4 Linear walks

The notion of linear walks, that is, words inducing non-zero tiles, and, conversely, words induced by traversals of non-zero tiles, conveys a relevant intuition of the link between tiles and two-way automata studied in the next section.

Definition 2.23 (Linear walk). A word $w \in (A \cup \bar{A})^*$ is a *linear walk*, or simply *walk* when $\theta(w) \neq 0$. The set of walks is denoted by $W(A)$.

Example 2.24. Since $\theta(\bar{a}abcba\bar{a}\bar{b}) = (a, bc, ba)$ (cf. the Introduction), the word $\bar{a}abcba\bar{a}\bar{b}$ is a walk. The word $a\bar{a}b$ is not a walk when $a \neq b$.

Remark 2.25. A word $w = a_1 \cdots a_n$ in $W(A)$, with $a_i \in A \cup \bar{A}$, can be viewed intuitively as a walk as follows. Consider the sequence of non-zero tiles $u_0, u_1, \ldots, u_n$ corresponding to the prefixes of w, i.e., $u_i = \theta(a_1 \cdots a_i)$.

This can be viewed as a walk that constructs the tile $u_n = \theta(w) \in T(A)$, starting from the unit tile $u_0 = (1, 1, 1)$. The "steps" in the walk are the transitions from the tile u_i to the tile u_{i+1}, for every $0 \leq i \leq n-1$. Note that $u_{i+1} = u_i \cdot \theta(a_{i+1})$. Thus, if $a_{i+1} = a \in A$, then u_{i+1} is obtained from u_i by moving the output root one letter to the right, after first adding a to the right of the output root, when there are no letters there. Similarly, if $a_{i+1} = \overline{a} \in A$, then the output root is moved one letter to the left but first a is added to its left, if necessary.

In this way, $\theta(w)$ is obtained from $(1, 1, 1)$ by moving the output root in a stepwise fashion. Since the input root does not change, w can be viewed as a walk that moves back and forth on the domain of $\theta(w)$, starting at the input root of $\theta(w)$ and ending at its output root. Note that the root path of u_i (i.e., its second component) equals $red(a_1 \cdots a_i)$ (easy proof by induction). In particular, the root path of $\theta(w)$ is $red(w)$.

As an example, the walk $\bar{a}abcba\bar{a}\bar{b}$ corresponds to the following sequence of tiles: $(1, 1, 1)$, $(a, \bar{a}, a)$, $(a, 1, 1)$, $(a, b, 1)$, $(a, bc, 1)$, $(a, bcb, 1)$, $(a, bcba, 1)$, (a, bcb, a), (a, bc, ba).

We finally note that the same sequence of tiles can be constructed when w is not in $W(A)$, but then $u_i = 0$ for some $i \geq 1$, and so $u_j = 0$ for all $j \geq i$.

Lemma 2.26. *If A is not a singleton, the set of walks over A is not a context-free subset of $(A \cup \bar{A})^*$.*

Proof. Let a and b be distinct letters of A, and let L be the intersection of the set of walks with the regular language $ba^*b\bar{b}\bar{a}^*\bar{b}ba^*b$. For all $m, n, p \geq 0$, the tile product

$$\theta(ba^mb) \cdot \theta(\bar{b}\bar{a}^n\bar{b}) \cdot \theta(ba^pb) = (1, ba^mb, 1) \cdot (ba^nb, \bar{b}\bar{a}^n\bar{b}, ba^nb) \cdot (1, ba^pb, 1)$$

is $(1, ba^nb, 1)$ if $m = n = p$, and 0 otherwise; thus $L = \{ba^nb\bar{b}\bar{a}^n\bar{b}ba^nb : n \geq 0\}$. Since L is not a context-free language, the set of walks is not context-free. □

More generally, we shall prove that walks *exactly* correspond to back-and-forth reading of words over A, thus relating walks with runs of a two-way automaton.

Definition 2.27 (The monoid of linear walks). Let $\perp = \theta^{-1}(0)$ be the set of words that are not walks. Clearly, $\perp$ is an ideal, i.e., $u \perp v \subseteq \perp$ for all $u, v \in (A \cup \bar{A})^*$.

The monoid of walks $W_0(A)$ is the Rees' quotient $W_0(A) = (A \cup \bar{A})^*/\perp$, that is, the monoid of walks $W_0(A)$ is defined by collapsing the set $\perp$ of non-walks into a single zero. In other words, $W_0(A) = W(A) \cup \{0\}$ with the product of two elements $u, v \in W_0(A)$ defined as uv when u, v and uv are in $W(A)$, and 0 otherwise.

Lemma 2.28. *Let $w \in (A \cup \bar{A})^*$ be a walk and let $u = (u_1, u_2, u_3) \in T^+(A)$ be a positive tile. If $\theta(w) = u$ then $w \simeq_W \bar{u}_1 u_1 u_2 u_3 \bar{u}_3$, and if $\theta(w) = u^{-1}$ then $w \simeq_W u_3 \bar{u}_3 \bar{u}_2 \bar{u}_1 u_1$.*

Proof. By induction on the length of w. If $w = 1$ then $\theta(w) = (1, 1, 1)$, and indeed we have $w \simeq_W 1$. Assume now that the property is true for some $w \in (A \cup \bar{A})^*$, and let $a \in A$.

Observe that if $u = u^{-1}$, i.e., $u_2 = 1$, then $\bar{u}_1 u_1 u_2 u_3 \bar{u}_3 = \bar{u}_1 u_1 u_3 \bar{u}_3 \simeq_W u_3 \bar{u}_3 \bar{u}_1 u_1 = u_3 \bar{u}_3 \bar{u}_2 \bar{u}_1 u_1$. In other words, when $u_2 = 1$, it suffices to prove one of the two statements of the lemma.

If wa is a walk, then so is w. We consider three different possible forms of $\theta(w)$, such that $\theta(w) \cdot (1, a, 1) \neq 0$.

Case 1: $\theta(w) = (u_1, u_2, 1) \in T^+(A)$ and $w \simeq_W \bar{u}_1 u_1 u_2$. Then $\theta(wa) = \theta(w) \cdot \theta(a) = (u_1, u_2 a, 1)$ and, since $\simeq_W$ is a congruence, we have $wa \simeq_W \bar{u}_1 u_1 u_2 a$.

Case 2: $\theta(w) = (u_1, u_2, au_3) \in T^+(A)$ and $w \simeq_W \bar{u}_1 u_1 u_2 a u_3 \bar{u}_3 \bar{a}$. Then $\theta(wa) = (u_1, u_2 a, u_3)$ and $wa \simeq_W \bar{u}_1 u_1 u_2 a u_3 \bar{u}_3 \bar{a} a$. By commutation of $u_3 \bar{u}_3$ and $\bar{a}a$, this implies that $wa \simeq_W \bar{u}_1 u_1 u_2 a \bar{a} a u_3 \bar{u}_3$. Simplifying $a\bar{a}a$ into a, we obtain $wa \simeq_W \bar{u}_1 u_1 u_2 a u_3 \bar{u}_3$.
Case 3: $\theta(w) = (u_1, au_2, u_3)^{-1}$ with $(u_1, au_2, u_3) \in T^+(A)$ and $w \simeq_W u_3 \bar{u}_3 \bar{u}_2 \bar{a} \bar{u}_1 u_1$. Then $\theta(wa) = (u_1 a, u_2, u_3)^{-1}$ and thus we have $wa \simeq_W u_3 \bar{u}_3 \bar{u}_2 \bar{a} \bar{u}_1 u_1 a$.

If $w\bar{a}$ is a walk, so is w and the proof is similar, again with three cases.
Case 1: $\theta(w) = (u_1, u_2 a, u_3)$ and $w \simeq_W \bar{u}_1 u_1 u_2 a u_3 \bar{u}_3$. Then $\theta(w\bar{a}) = (u_1, u_2, au_3)$ and $w\bar{a} \simeq_W \bar{u}_1 u_1 u_2 a u_3 \bar{u}_3 \bar{a}$.
Case 2: $\theta(w) = (1, u_2, u_3)^{-1}$ and $w \simeq_W u_3 \bar{u}_3 \bar{u}_2$. Then $\theta(w\bar{a}) = (1, au_2, u_3)^{-1}$ and$w\bar{a} \simeq_W u_3 \bar{u}_3 \bar{u}_2 \bar{a}$.
Case 3: $\theta(w) = (u_1 a, u_2, u_3)^{-1}$ and $w \simeq_W u_3 \bar{u}_3 \bar{u}_2 \bar{a} \bar{u}_1 u_1 a$. Then $\theta(w\bar{a}) = (u_1, au_2, u_3)^{-1}$ and we have $w\bar{a} \simeq_W u_3 \bar{u}_3 \bar{u}_2 \bar{a} \bar{u}_1 u_1 a \bar{a}$. By commutation of $\bar{u}_1 u_1$ and $a\bar{a}$ we have$w\bar{a} \simeq_W u_3 \bar{u}_3 \bar{u}_2 \bar{a} a \bar{a} \bar{u}_1 u_1$. Simplifying $\bar{a}a\bar{a}$ into $\bar{a}$ we obtain $w\bar{a} \simeq_W u_3 \bar{u}_3 \bar{u}_2 \bar{a} \bar{u}_1 u_1$. □

Corollary 2.29. *Let $w_1, w_2 \in W(A)$ be two walks. Then $\theta(w_1) = \theta(w_2)$ if, and only if, $w_1 \simeq_W w_2$.*

Proof. If: by Lemma 2.15. *Only if*: follows from Lemma 2.28. Indeed, if $\theta(w_1) = \theta(w_2)$ is a positive tile (u_1, u_2, u_3), then we have $w_1 \simeq_W \bar{u}_1 u_1 u_2 u_3 \bar{u}_3 \simeq_W w_2$, and if $\theta(w_1) = \theta(w_2)$ is a negative tile $(u_1, u_2, u_3)^{-1}$, then we have $w_1 \simeq_W u_3 \bar{u}_3 \bar{u}_2 \bar{u}_1 u_1 \simeq_W w_2$. □

In other words, the previous results show that the diagram depicted in Figure 6 commutes.

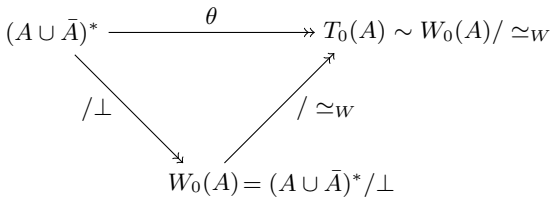


Fig. 6. The walks and tiles diagram

Corollary 2.29 proved above gives a characterization of θ on walks. It remains to understand the structure of $\perp = \theta^{-1}(0)$.

Lemma 2.30. *Let $w \in (A \cup \bar{A})^*$. We have $\theta(w) = 0$ if, and only if, there exists $u \in (A \cup \bar{A})^*$ such that $u \simeq_W w$ and u has a factor of the form $a\bar{b}$ or $\bar{a}b$ for two distinct letters $a, b \in A$.*

Proof. If: by Lemma 2.15. *Only if*: assume that $\theta(w) = 0$. Let then w_1 be the longest prefix of w such that $\theta(w_1) \neq 0$.

We first consider the case when $w = w_1 b w_2$ with $b \in A$. Since $\theta(w_1 b) = 0$, there are two subcases with $u_1, u_2, u_3 \in A^*$ and $a \in A$.
Case 1: $\theta(w_1) = (u_1, u_2, au_3)$ with $b \neq a$. By Lemma 2.28, we have $w_1 \simeq_W \bar{u}_1 u_1 u_2 a u_3 \bar{u}_3 \bar{a}$ hence, since $w = w_1 b w_2$, we have $w \simeq_W \bar{u}_1 u_1 u_2 a u_3 \bar{u}_3 \bar{a} b w_2$ which has the factor $\bar{a}b$.
Case 2: $\theta(w_1) = (u_1, au_2, u_3)^{-1}$ with $b \neq a$. By Lemma 2.28, we have $w_1 \simeq_W u_3 \bar{u}_3 \bar{u}_2 \bar{a} \bar{u}_1 u_1$ hence $w \simeq_W u_3 \bar{u}_3 \bar{u}_2 \bar{a} \bar{u}_1 u_1 b w_2$ hence, by expanding $\bar{a}$ into $\bar{a}a\bar{a}$, $w \simeq_W u_3 \bar{u}_3 \bar{u}_2 \bar{a} a \bar{a} \bar{u}_1 u_1 b w_2$, and thus, by commutation of $a\bar{a}$ and $\bar{u}_1 u_1$, we have $w \simeq_W u_3 \bar{u}_3 \bar{u}_2 \bar{a} \bar{u}_1 u_1 a \bar{a} b w_2$ which has the factor $\bar{a}b$.

Similarly we consider the case when $w = w_1 \bar{b} w_2$ with $b \in A$. Again there are two subcases.
Case 1: $\theta(w_1) = (u_1, u_2 a, u_3)$ with $b \neq a$. By Lemma 2.28, we have $w_1 \simeq_W \bar{u}_1 u_1 u_2 a u_3 \bar{u}_3$ hence

$w \simeq_W \bar{u}_1 u_1 u_2 a u_3 \bar{u}_3 \bar{b} w_2$. It follows that $w \simeq_W \bar{u}_1 u_1 u_2 a \bar{a} a u_3 \bar{u}_3 \bar{b} w_2$ by expanding a into $a\bar{a}a$. Then, by commutation of $\bar{a}a$ with $u_3\bar{u}_3$ we have $w \simeq_W \bar{u}_1 u_1 u_2 a u_3 \bar{u}_3 \bar{a} a \bar{b} w_2$ which has the factor $a\bar{b}$.
Case 2: $\theta(w_1) = (u_1 a, u_2, u_3)^{-1}$ with $b \neq a$. By Lemma 2.28, we have $w_1 \simeq_W u_3 \bar{u}_3 \bar{u}_2 \bar{a} \bar{u}_1 u_1 a$ hence $w \simeq_W u_3 \bar{u}_3 \bar{u}_2 \bar{a} \bar{u}_1 u_1 a \bar{b} w_2$ which has the factor $a\bar{b}$. □

We conclude our study of the monoid of tiles by proving that the monoid $T_0(A)$ is isomorphic to the monoid of McAlister [37, 34], that is, some Rees' quotient of the free inverse monoid $FIM(A)$ generated by A.

Let $\simeq_M$ be the McAlister congruence, defined as the least congruence in the extended monoid $(A \cup \bar{A})^* \cup \{0\}$ such that $\simeq_W \subseteq \simeq_M$ and

$$a\bar{b} \simeq_M 0 \text{ and } \bar{a}b \simeq_M 0$$

for all distinct $a, b \in A$. Let $M_A = (A \cup \bar{A})^* \cup \{0\}/\simeq_M$ be the monoid of McAlister.

Theorem 2.31. *The monoid of tiles $T_0(A)$, the quotient of walks $W_0(A)/\simeq_W$ by the Wagner congruence and the monoid of McAlister are isomorphic.*

Proof. By Corollary 2.29, $T_0(A)$ is isomorphic to $W_0(A)/\simeq_W$. By Corollary 2.29 and Lemma 2.30, the congruence associated with the morphism θ (extended to $(A\cup\bar{A})^*\cup\{0\}$ by $\theta(0) = 0$) coincides with the McAlister congruence. □

This situation is summarized by the commuting diagram depicted in Figure 7 below where

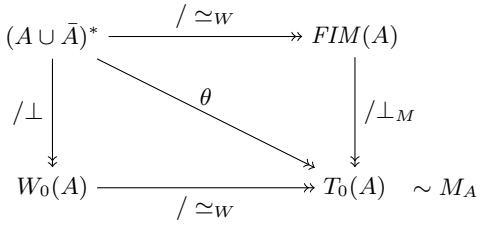


Fig. 7. The walks, tiles and birooted trees diagram

the ideal of $FIM(A)$ generated by elements of the form $a\bar{b}$ and $\bar{a}b$ with $a \neq b$ is denoted $\perp_M$.

Remark 2.32. In some sense, Lemma 2.28 captures most of the combinatorial analysis of two-way automata runs made in [39, 5]. More precisely, according to Pécuchet's definition [39], a word bisection is any quadruple of words

$$((u_1, u_2), (v_1, v_2)) \in (A^* \times A^*) \times (A^* \times A^*)$$

such that $u_1 u_2 = v_1 v_2$. Then, one can check that the mapping that maps every non-zero tile $(u_1, u_2, u_3) \in T_0(A)$ to the quadruple

$$((u_1, u_2 \cdot u_3), (u_1 \cdot u_2, u_3)) \in (A^* \times A^*) \times (A^* \times A^*)$$

is a well-defined bijection from non-zero tiles to word bisections. In other words, Pécuchet's approach somehow provides a fourth definition of McAlister monoid. Note that this link with McAlister's monoid, defined in [37] but emphasized in [34], was, at best, left implicit in Pécuchet's

and Birget's works [39, 5], even though some connections with the theory of inverse semigroups are made. It must be noticed, however, that in this paper, we are more interested in *what* two-way automata read than in *how* they perform readings, which was Pécuchet's and Birget's main interest. A similar observation can be made about the study of zig-zag codes [2, 8, 35] where no connection with McAlister monoid is mentioned.

3 Regular languages of tiles and two-way automata

Given an alphabet A assumed to have at least two letters, a *language of tiles* on the alphabet A is any subset $L \subseteq T(A)$ of *non-zero* tiles. The consequences of such a restriction, essentially harmless, are discussed in some remarks below.

In this section, we study the class *REC* of recognizable tile languages, that is, languages definable by means of monoid morphisms into finite monoids, and the class *REG* of regular languages, that is, languages definable by means of regular expressions.

The class *REC* is characterized and shown to be strictly included in the class *REG*. The class *REG* is shown to correspond to the class of tile languages definable by means of finite state two-way automata which semantics is simply extended to tiles.

3.1 Regular languages of tiles

We define in this section the class *REG* of regular languages of tiles. It follows the classical definition induced by the monoid structure of $T_0(A)$ up to the fact that we restrict to languages of non-zero tiles. In particular, the product of languages is restricted to non-zero products of tiles. The intended meaning of this restriction is the following: when two languages of tiles model the possible behaviors of two processes, the product of these languages models their sequential composition. The processes implicitly communicate by agreeing on compatible behaviors, that is, by restricting to non-zero products.

Formally, we define the following operations on languages of non-zero tiles:

- ▷ addition as union: $M + N = M \cup N$
- ▷ multiplication: $M \cdot N = \{u \cdot v \in T(A) : u \in M, v \in N\}$
- ▷ star: $M^* = \bigcup_{n \geq 0} M^n$ with $M^0 = \{(1, 1, 1)\}$ and $M^{k+1} = M \cdot M^k$ for every $k \in \mathbb{N}$

The next lemma states that these operations satisfy the usual properties of the operations on word languages (the same proofs apply).

Lemma 3.1. *For all M, N and $P \subseteq T(A)$:*

- ▷ *$M \cdot (N + P) = M \cdot N + M \cdot P$ and $(M + N) \cdot P = M \cdot P + N \cdot P$*
- ▷ *$M^* \cdot N$ is the least solution (with respect to inclusion) of the equation $X = M \cdot X + N$.*

Regular languages of tiles are defined by the usual notion of regular expression.

Definition 3.2 (Regular language of tiles). A language $M \subseteq T(A)$ of tiles is *regular* if it can be defined as the result of finitely many additions, multiplications and star operations over finite languages of (non-zero) tiles. The class of regular languages of tiles is denoted by *REG*.

Example 3.3. The set of non-zero tiles is regular since $T(A) = (\theta(A \cup \bar{A}))^*$. We prove later (Lemma 3.8) that the set $E(A)$ of non-zero idempotent tiles is not regular.

The inverse monoid structure of $T_0(A)$ induces three more operations on languages of tiles:

- ▷ inverse: $M^{-1} = \{u^{-1} \in T(A) : u \in M\}$,
- ▷ right projection: $M^R = \{u \cdot u^{-1} \in T(A) : u \in M\}$,
- ▷ left projection: $M^L = \{u^{-1} \cdot u \in T(A) : u \in M\}$.

The following identities are straightforward.

Lemma 3.4. *For all $M, N \subseteq T(A)$:*

- ▷ $(M + N)^{-1} = M^{-1} + N^{-1}$
- ▷ $(M \cdot N)^{-1} = N^{-1} \cdot M^{-1}$
- ▷ $(M^*)^{-1} = (M^{-1})^*$

As a consequence, the class *REG* is closed under inverse. On the contrary, we shall see (Lemma 3.7) that the class *REG* is not closed under left and right projections.

The following characterization is a variation of a well-known result for rational subsets of monoids (see, e.g., Proposition III.2.2 of [3]).

Theorem 3.5. *A language of tiles $X \subseteq T(A)$ is regular if, and only if, there exists a regular language of words $L \subseteq (A \cup \bar{A})^*$ such that $X = \theta(L \cap W(A))$.*

Proof. We first observe that, for every language $L \subseteq (A \cup \bar{A})^*$ we have $\theta(L \cap W(A)) = \theta(L) - \{0\}$. Then, the proof goes by induction on the structure of regular expressions, using that θ is a monoid morphism.

If. Let $L \subseteq (A \cup \bar{A})^*$ be a regular language of words. If L is finite, then $\theta(L) - \{0\}$ is a finite tile language. It follows that any regular expression for L, thanks to our definition of sum, product and star of tile languages, can be reinterpreted over languages of tiles as a regular expression for $\theta(L) - \{0\}$. Note in particular that, for all tile languages M and N, we have $M \cdot N = \{u \cdot v \in T_0(A) : u \in M, v \in N\} - \{0\}$.

Only if. Let $X \subseteq T(A)$ be a regular language of tiles. If X is finite, then $X = \theta(F) - \{0\}$ where $F = \{\overline{u}_1 u_1 u_2 u_3 \overline{u}_3 : (u_1, u_2, u_3) \in X\}$ is a finite language. Then, any regular expression for X can be interpreted over languages of words of $(A \cup \bar{A})^*$ denoting then a language L such that $X = \theta(L) - \{0\}$. □

Remark 3.6. Following the definitions of [3], a subset of a monoid M is *rational* if it can be defined as the result of finitely many additions, multiplications and star operations over finite subsets of M. By Proposition III.2.2 of [3], a set Y of tiles is a rational subset of $T_0(A)$ if and only if there exists a regular language of words $L \subseteq (A \cup \bar{A})^*$ such that $Y = \theta(L)$. In that case, $L \cup \{\bar{a}b\}$ is also regular, and $Y \cup \{0\} = \theta(L \cup \{\bar{a}b\})$ (where $a, b \in A, a \neq b$). Thus, by Theorem 3.5, a language of tiles $X \subseteq T(A)$ is regular if and only if $X \cup \{0\}$ is a rational subset of the monoid $T_0(A)$. Note also that, by Theorem 3.5 again, every rational language of tiles $X \subseteq T(A)$ is regular because if $X = \theta(L)$, since $0 \notin X$ then $L \subseteq W(A)$.

The following lemma gives a first example of a simple non-regular language of tiles.

Lemma 3.7. *The right projection M^R of the regular tile language $M = \theta(A^*)$ is not regular. As a consequence, the class of regular languages of tiles is not closed under right (or left) projection.*

Proof. Let M and M^R be the languages defined as above, that is, $M^R = \{(1, 1, y) \in T(A) : y \in A^*\}$. Moreover, let $a \in A$. We first observe that since every tile in M^R is idempotent, i.e., $M^R \subseteq E(A)$, any word $w \in (A \cup \bar{A})^*$ such that $\theta(w) \in M^R$ must have the same number of a's and of $\bar{a}$'s, which we denote by $|w|_a = |w|_{\bar{a}}$ (this is because $red(w) = 1$, see also Remark 2.25 that gives some more intuition on the notion of walks).

Assume now that M^R is regular. By Theorem 3.5, there exists a regular language $L \subseteq (A \cup \bar{A})^*$ such that $M^R = \theta(L \cap W(A))$. Let $\mathcal{A}$ be a deterministic finite-state automaton recognizing L. Let N denote the number of states of $\mathcal{A}$, and let $n > N$. Then there exists a word $u \in L \cap W(A)$ such that $\theta(u) = (1, 1, a^n) \in M^R$. Clearly, $u \in \{a, \bar{a}\}^*$ (see Remark 2.25).

We easily prove by induction (see again Remark 2.25) that for every k such that $0 \leq k \leq n$, there exists a prefix v_k of u such that $\theta(v_k) = (1, a^k, 1)$. Moreover, taking the least prefix that satisfies this property, the sequence of prefixes $v_0, v_1, \ldots, v_n$ of u is totally ordered by the prefix order.

But since $n > N$, there necessarily exist two prefixes v_{k_1} and v_{k_2}, with $k_1 < k_2$, such that the runs of the automaton $\mathcal{A}$ starting in the initial state and reading v_{k_1} and v_{k_2} reach the same state. Since v_{k_1} is a prefix of v_{k_2} and v_{k_2} is a prefix of u, there are words $w_1, w_2 \in \{a, \bar{a}\}^*$ such that $v_{k_2} = v_{k_1} w_1$ and $u = v_{k_2} w_2$. It follows that $\mathcal{A}$ accepts the word $u' = v_{k_1} w_2$. Since $u' \in \{a, \bar{a}\}^* \subseteq W(A)$ (see Remark 2.5), we have that $u' \in L \cap W(A)$ and so $\theta(u') \in M^R$ and $|u'|_a = |u'|_{\bar{a}}$. Now, the equality $v_{k_2} = v_{k_1} w_1$ implies that $|w_1|_a - |w_1|_{\bar{a}} = k_2 - k_1$ (see again Remark 2.25). However, by construction, for every $x \in \{a, \bar{a}\}$, we have $|u|_x = |u'|_x + |w_1|_x$. Since $|u|_a = |u|_{\bar{a}}$, we thus have $|u'|_{\bar{a}} - |u'|_a = |w_1|_a - |w_1|_{\bar{a}} = k_2 - k_1 > 0$: a contradiction.

A similar proof shows that M^L is not regular. □

The next lemma gives a second example of a simple non-regular tile language.

Lemma 3.8. *The set $E(A)$ of non-zero idempotent tiles is not regular.*

Proof. The proof is exactly the same as that of the previous lemma, with M^R replaced by $E(A)$. □

Last, we give an example of a regular tile language that is not rational (see Remark 3.6).

Lemma 3.9. *The regular tile language $T(A)$ of non zero tiles is not rational.*

Proof. The proof is very similar to the one of Lemma 3.7. Assume that $T(A)$ is rational. By Remark 3.6, there exists a regular language $L \subseteq (A \cup \bar{A})^*$ such that $T(A) = \theta(L)$. Let $\mathcal{A}$ be a deterministic finite-state automaton recognizing L, and let n be larger than the number of states of $\mathcal{A}$. Let $a, b \in A$ with $a \neq b$. Then there exists a word $u \in L$ such that $\theta(u) = (1, 1, ba^n) \in T(A)$. As in the proof of Lemma 3.7, there exist words v_{k_1}, v_{k_2}, with $k_1 < k_2$, such that $\theta(v_{k_1}) = (1, ba^{k_1}, 1)$, $\theta(v_{k_2}) = (1, ba^{k_2}, 1)$ and the runs of $\mathcal{A}$ on v_{k_1} and v_{k_2} reach the same state. Moreover, there exist words w_1, w_2 such that $v_{k_2} = v_{k_1} w_1$ and $u = v_{k_2} w_2$. It follows that $\mathcal{A}$ accepts the word $u' = v_{k_1} w_2$, and so $u' \in L$ and $\theta(u') \in T(A)$. Now, the equality $u = v_{k_2} w_2$ implies that $red(w_2) = \bar{a}^{k_2} \bar{b}$ (see Remark 2.25). However, that implies that $red(u') = red(red(v_{k_1}) red(w_2)) = red(ba^{k_1} \bar{a}^{k_2} \bar{b}) = b\bar{a}^{k_2 - k_1} \bar{b}$, which is not in $A^* \cup \bar{A}^*$, contradicting the fact that $\theta(u')$ is a non-zero tile. □

3.2 Recognizable vs regular languages of tiles

For the sake of completeness, we briefly review here the notion of algebraic recognizability when applied to languages of tiles and relate it to the notion of regular languages of words.

Definition 3.10 (Recognizable languages). A language of tiles $L \subseteq T(A)$ is *recognizable* when there exist a monoid M, a monoid morphism $\varphi : T_0(A) \to M$, and a finite subset F of M, such that $L = \varphi^{-1}(F)$; or, equivalently, when the syntactic congruence defined by $u \sim_L v \iff \forall x, y \in T_0(A)\ (x \cdot u \cdot y \in L \Leftrightarrow x \cdot v \cdot y \in L)$ is of finite index. The class of recognizable languages of tiles is denoted by *REC*.

The following characterization is analogous to the equivalent result over the free inverse monoid (Lemma 3.1 of [46]). It follows from classical arguments concerning recognizable languages.

Theorem 3.11. *A language $L \subseteq T(A)$ of tiles is recognizable if and only if $\theta^{-1}(L)$ is a regular language of words.*

Proof.If: let $K = \theta^{-1}(L)$, and let $\sim_K$ denote the syntactic congruence of K in $(A \cup \bar{A})^*$. For all $u, v \in (A \cup \bar{A})^*$, we have $u \sim_K v \Rightarrow \theta(u) \sim_L \theta(v)$: indeed, since θ is onto, for all $x, y \in T_0(A)$ there exist $\alpha, \beta \in (A \cup \bar{A})^*$ such that $\theta(\alpha) = x$ and $\theta(\beta) = y$; if $u \sim_K v$ we have $x \cdot \theta(u) \cdot y \in L \Rightarrow \theta(\alpha u \beta) \in L \Rightarrow \alpha u \beta \in K \Rightarrow \alpha v \beta \in K \Rightarrow x \cdot \theta(v) \cdot y \in L$ and by symmetry, we have $x \cdot \theta(v) \cdot y \in L \Rightarrow x \cdot \theta(u) \cdot y \in L$.
Thus $[u]_{\sim_K} \mapsto [\theta(u)]_{\sim_L}$ is a well-defined mapping of $(A \cup \bar{A})^*/\sim_K$ onto $T_0(A)/\sim_L$. If K is a regular language of words then $(A \cup \bar{A})^*/\sim_K$ is finite and thus $\sim_L$ is of finite index.
Only if: let $\varphi : T_0(A) \to M$ be a monoid morphism, let F be a finite subset of M and let $L = \varphi^{-1}(F)$. Then $\psi = \varphi \circ \theta$ is a monoid morphism from $(A \cup \bar{A})^*$ to M, and $\theta^{-1}(L) = \psi^{-1}(F)$ is recognized (as a language of words) by the monoid M and the morphism ψ. □

Corollary 3.12. *Every recognizable tile language is regular.*

Proof. Let $L \subseteq T(A)$ be a recognizable tile language. Then $M = \theta^{-1}(L)$ is a regular language of words by Theorem 3.11. Since θ is surjective, $L = \theta(M)$. So, since $M \subseteq W(A)$, $\theta(M)$ is regular by Theorem 3.5. □

Note that, by the proof of this corollary, every recognizable tile language is even rational (see Remark 3.6). But not all regular languages are recognizable:

Lemma 3.13. *The regular tile language $L = \theta(\bar{a}^*\bar{b}ba^*)$ is not recognizable.*

Proof. Since $\bar{a}^*\bar{b}ba^* \subseteq W(A)$, by Theorem 3.5, the language L is regular. We show that the syntactic congruence $\sim_L$ is of infinite index. For all $m, k \in \mathbb{N}$ let $u_m = \theta(\bar{a}^m\bar{b}ba^m) = (ba^m, 1, 1) \in L$ and let $v_k = (a^k, 1, 1)$. Then $u_m \cdot v_k = u_m$ if $k \leq m$, and $u_m \cdot v_k = 0$ if $k > m$. Hence $u_m \cdot v_k \in L$ if and only if $k \leq m$. Assume now that $u_m \sim_L u_n$ for some $m, n \in \mathbb{N}$. Then for every $k \in \mathbb{N}$, we have $k \leq m$ if and only if $k \leq n$, hence we necessarily have $m = n$. □

By Corollary 3.12 and Lemma 3.13 we have

Corollary 3.14. *$REC \subsetneq REG$.*

Remark 3.15. We have already seen that $T(A)$ is regular while $W(A) = \theta^{-1}(T(A))$ is not even context-free (Lemma 2.26). It follows that $T(A)$ is also a regular language that is not recognizable. Note that, as proved in Lemma 3.9, $T(A)$ is not even rational. Thus, since the language L of Lemma 3.13 is rational, the class of rational tile languages (see Remark 3.6) lies properly between the classes of recognizable and regular tile languages.

Though some simple regular languages are not recognizable, the class *REC* does contain non-trivial languages of tiles. In [23], it is shown that recognizable languages are strongly related with bi-infinite periodic words. We refer the reader to this presentation for a combinatorial characterization of these languages. In this paper, we just give an example of a non-trivial recognizable language, that derives from the bi-infinite word ${}^{\omega}(ab)(ab)^{\omega}$.

Example 3.16. Let $M = \{0, 1, (a, a), (a, b), (b, a), (b, b)\}$, and let $\odot$ be the product defined over M by

$$\forall x, y, z, t \in \{a, b\}, \quad (x, y) \odot (z, t) = \begin{cases} 0 \text{ if } y \neq z \\ (x, t) \text{ if } y = z \end{cases}$$

with 1 neutral and 0 absorbing.

We first observe that $(M, \odot)$ is a monoid. It is well known that partial bijections over a given set form a monoid with composition as product. Then it suffices to check that $(M, \odot)$ corresponds to the submonoid of partial bijections over the set $\{a, b\}$ generated by the partial bijection $m_{a,b}$ that maps a to b (encoded by (a, b)) and the partial bijection $m_{b,a}$ that maps b to a (encoded by (b, a)). Indeed, the additional partial bijections obtained from the generators are $m_{a,a} = m_{a,b} \circ m_{b,a}$ (encoded by (a, a)) and $m_{b,b} = m_{b,a} \circ m_{a,b}$ (encoded by (b, b)). The empty bijection 0 is given, for instance, by $0 = m_{a,a} \circ m_{b,b}$.

Incidentally, we also observe that $(M, \odot)$ is even an inverse monoid. Indeed, it is also well-know in inverse semigroup theory (see Wagner Theorem in [33]) that the monoid of partial bijections over a set is an inverse monoid with function inverses as monoid inverses, and the two generators of M are inverses of each other.

Now, let $\varphi : T_0(\{a, b\}) \to M$ be the mapping defined as follows. Let $u \in T_0(\{a, b\})$. If $u \in \{0, 1\}$ then $\varphi(u) = u$. Otherwise, if the domain of u is not a factor of the infinite word $(ab)^\omega$, then $\varphi(u) = 0$. Otherwise, $\varphi(u) = (x, y)$ where x (resp. y) is the letter directly to the right of the input (resp. the output root) of u embedded in $(ab)^\omega$: the pair (x, y) can be seen as the partial bijection $m_{x,y}$ induced by (the embedded) tile u that maps x (right after its input root) to y (right after its output root).

For example, $\varphi((a, ab, 1)) = 0$ because aab is not a factor of $(ab)^\omega$, $\varphi((b, aba, 1)) = (a, b)$ because any factor $baba$ of $(ab)^\omega$ is followed by b, and $\varphi((b, 1, 1)) = (a, a)$. Since the above partial maps compose well, it should be clear that φ is a monoid morphism, i.e., $\varphi(u \cdot v) = \varphi(u) \odot \varphi(v)$ (see also the discussion before Theorem 2.8).

Then, given $L_1 = (ab)^* + b(ab)^*$, $L_2 = (ab)^*$, and $L_3 = (ab)^* + (ab)^*a$, we observe that the tile language $L_1 \times L_2 \times L_3$ is recognizable since it is precisely the inverse image of the set $\{1, (a, a)\} \subseteq M$ under φ.

3.3 Two-way automata

We prove here that the regular languages of tiles are the languages recognized by finite-state two-way automata. Our proposed definition, especially when extended with pebbles as in the next section, is inspired by the notion of stack automata studied in [15].

Definition 3.17 (Two-way automaton). A *finite-state two-way automaton* over an alphabet A is defined as a standard finite-state automaton over $A \cup \bar{A}$, i.e., a quadruple $\mathcal{A} = \langle Q, I, F, \Delta \rangle$ with a finite set of states Q, a set of initial states $I \subseteq Q$, a set of final states $F \subseteq Q$, and a transition table $\Delta : (A \cup \bar{A}) \to \mathcal{P}(Q \times Q)$.

A *run* of $\mathcal{A}$ over a word of $(A \cup \bar{A})^*$ is a finite sequence $\rho = q_0 a_1 q_1 \cdots q_{n-1} a_n q_n$ where $n \geq 0$, $q_0, \ldots, q_n \in Q$ and $a_1, \ldots, a_n \in A \cup \bar{A}$, such that for every $1 \leq i \leq n$, we have $(q_{i-1}, q_i) \in \Delta(a_i)$.

The run ρ is *accepting* if $q_0 \in I$ and $q_n \in F$. In that case, we say that the associated word $a_1 \cdots a_n \in (A \cup \bar{A})^*$ is accepted by the automaton $\mathcal{A}$.

Let $L(\mathcal{A})$ denote the set of words over the alphabet $A \cup \bar{A}$ that are accepted by $\mathcal{A}$, and let $L_W(\mathcal{A}) = L(\mathcal{A}) \cap W(A)$ denote the language of *walks* recognized by $\mathcal{A}$.

Remark 3.18. Observe that $L_W(\mathcal{A})$ may not be regular, since the trivial automaton $\mathcal{A}$ such that $L(\mathcal{A}) = (A \cup \bar{A})^*$ recognizes the language of walks $L_W(\mathcal{A}) = W(A)$, which by Lemma 2.26 is not even context-free.

The language of *tiles* recognized by $\mathcal{A}$ is defined as the set $L_T(\mathcal{A}) = \theta(L(\mathcal{A}) \cap W(A))$.

Remark 3.19. A more classical definition of $L_T(\mathcal{A})$ can be obtained by viewing the elements of $Q \times T_0(A)$ as configurations of $\mathcal{A}$, and by defining a binary computation relation $\Rightarrow_{\mathcal{A}}$ on $Q \times T_0(A)$ as follows: $(q,u) \Rightarrow_{\mathcal{A}} (q',u')$ if there exists $a \in A \cup \bar{A}$ such that $(q,q') \in \Delta(a)$ and $u' = u \cdot \theta(a)$. It is straightforward to prove that $L_T(\mathcal{A})$ is the set of non-zero tiles u such that $(q,1) \Rightarrow^*_{\mathcal{A}} (q',u)$ for some $q \in I$ and $q' \in F$.

As suggested in Remark 2.25, a computation $(q_0,1) \Rightarrow_{\mathcal{A}} (q_1,u_1) \Rightarrow_{\mathcal{A}} \cdots \Rightarrow_{\mathcal{A}} (q_n,u_n)$ of the automaton $\mathcal{A}$ can then be viewed as a walk back and forth on the domain of u_n. For every i, $0 \leq i \leq n$, the output root of u_i can be viewed as the position of the reading head of $\mathcal{A}$ (*between* letters of the domain), wheras the input root of u_i is the original position of the reading head at the start of the computation. For every $a \in A$, if (in the above computation step) $u' = u \cdot \theta(a)$, then $\mathcal{A}$ reads the letter a from left to right, whereas if $u' = u \cdot \theta(\bar{a})$, then it reads a from right to left.

The language of *words* $L_S(\mathcal{A}) \subseteq A^*$ recognized by the two-way automaton $\mathcal{A}$ is then defined by $L_S(\mathcal{A}) = \{u \in A^* : (1,u,1) \in L_T(\mathcal{A})\}$. It corresponds to the usual notion of language of words recognized by a two-way automaton.

Remark 3.20. Our definitions of tile languages and tile languages product provide a fairly simple relationship between the word language and the tile language recognized by two-way automata. Indeed, given an additional letter $\#$ such that $\# \notin A$, we easily check that

$$\theta(\#) \cdot L_T(\mathcal{A}) \cdot \theta(\#) = \theta(\#) \cdot \theta(L_S(\mathcal{A})) \cdot \theta(\#) = \{1\} \times \#L_S(\mathcal{A})\# \times \{1\}$$

for every two-way automaton $\mathcal{A}$. In other words, the product by $\theta(\#)$ both on the left and on the right of the tile language $L_T(\mathcal{A})$ recognized by $\mathcal{A}$ acts as a filter that selects the word language $L_S(\mathcal{A})$ recognized by $\mathcal{A}$. This observation is used below to recover a proof of Shepherdson's theorem (Corollary 4.9).

Kleene's theorem extends to languages of tiles. Let *2WA* denote the class of tile languages recognized by finite-state two-way automata:

Theorem 3.21. *REG = 2WA.*

Proof. If: Let $X \subseteq T(A)$ be a regular language of tiles. By Theorem 3.5, there is a regular language of words $L \subseteq (A \cup \bar{A})^*$ such that $X = \theta(L \cap W(A))$. By Kleene's theorem, there exists a finite-state automaton $\mathcal{A}$ such that $L(\mathcal{A}) = L$ henceforth $L_T(\mathcal{A}) = X$.

Only if: Conversely, let $\mathcal{A}$ be a finite-state two-way automaton. By Kleene's theorem $L(\mathcal{A})$ is regular, hence, by Theorem 3.5, the language $L_T(\mathcal{A}) = \theta(L(\mathcal{A}) \cap W(A))$ is regular. □

4 Beyond regular languages

In Formal Language Theory, definability in Monadic Second-Order Logic is a typical yardstick of expressiveness which can be defined independently of the underlying algebraic structures [47]. We quickly review how languages of tiles can be defined by MSO formulas, and we characterize MSO-definable languages of tiles by expressions involving regular languages of words.

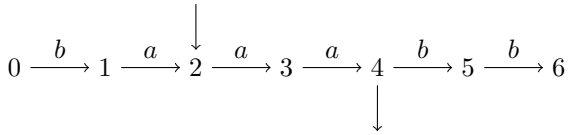
We extend the notion of regular expression using a projection operator onto languages of idempotent tiles:

Definition 4.1 (Idempotent projection). For every language $M \subseteq T(A)$ of tiles, let $M^E = M \cap E(A)$ denote the *idempotent projection* of M.

The nesting depth of idempotent projection operators induces a hierarchy of k-regular expressions that is shown equivalent to the hierarchy of k-pebble automata. We show that all k-regular languages are MSO-definable, and that MSO-definable languages correspond to 1-regular languages, thus proving that the hierarchy collapses at level $k = 1$.

4.1 MSO-definable languages

Any positive non-zero tile $u \in T(A)$ can be seen as an FO-structure on the signature $\{R_a\}_{a\in A}$ of binary relation symbols, extended with two constants *in* and *out* for the input and output roots. For instance, the triple $u = (ba, aa, bb)$ depicted in the following picture



can be seen as the FO-structure t_u over $dom(t_u) = \{0, 1, 2, 3, 4, 5, 6\}$, with relations R_a and R_b given by $R_a = \{(1, 2), (2, 3), (3, 4)\}$ and $R_b = \{(0, 1), (4, 5), (5, 6)\}$, and constants $in(t_u) = 2$ and $out(t_u) = 4$. Any negative non-zero tile of the form $u = (u_1u_2, \overline{u}_2, u_2u_3)$ can also be seen, as depicted in all examples so far, as the FO-structure obtained from the one of the tile (u_1, u_2, u_3) just by interchanging the constants *in* and *out*.

Then, a language $L \subseteq T(A)$ is *MSO-definable* when there is an MSO formula of the form $\varphi(U, x, y)$ where U is a set variable and x and y are two FO-variables such that, for every $u \in T(A)$, we have $u \in L$ if and only if $t_u \models \varphi(dom(t_u), in(t_u), out(t_u))$.

Remark 4.2. Without loss of generality, we assume that all quantifications in such a formula $\varphi(U, x, y)$ are relativized with respect to U, that is, in every sub-formula of the form $\forall z\, \psi$ (resp. $\exists z\, \psi$), we assume that ψ is of the form $z \in U \Rightarrow \psi'$ (resp. $z \in U \wedge \psi'$) and similarly for quantified set variables $Z \subseteq U$. This standard technique, called *relativization*, allows us to talk about submodels of a given model. Moreover, to ensure that these submodels are well defined, we assume that the formula $\varphi(U, x, y)$ checks that the set U is connected (viewing t_u as a graph) and that both x and y belong to U.

Under these assumptions, for all $X \subseteq dom(t_u)$ and $x_1, x_2 \in dom(t_u)$, we have$t_u \models \varphi(X, x_1, x_2)$ if, and only if, there exists a (unique) tile v in the tile language L defined by $\varphi(U, x, y)$ such that t_v is (up to isomorphism) the FO-structure (on the same signature) consisting of the subgraph of t_u induced by $X \subseteq dom(t_u)$, with x_1 and x_2 as the values of *in* and *out*, respectively. We will say that v (determined by X, x_1 and x_2) is a "local" tile of u belonging to L and connecting x_1 to x_2.

We now prove several closure properties of the class *MSO* of MSO-definable languages of tiles with an associated corollary.

Theorem 4.3 (Robustness). *For all MSO-definable languages of tiles $M, N \subseteq T(A)$, the languages $M \cup N$, $M \cdot N$, M^*, M^{-1}, M^L, M^R and M^E are also MSO-definable.*

Proof. Let $\varphi_M(U, x, y)$ and $\varphi_N(U, x, y)$ be two MSO formulas respectively defining M and N.

Union: take $\varphi_{M\cup N}(U, x, y) \equiv \varphi_M(U, x, y) \vee \varphi_N(U, x, y)$.

Inverse: take $\varphi_{M^{-1}}(U, x, y) \equiv \varphi_M(U, y, x)$.

Right projection: take $\varphi_{M^R}(U, x, y) \equiv (x = y) \wedge \exists z\, \varphi_M(U, x, z)$.

Left projection: take $\varphi_{M^L}(U, x, y) \equiv (x = y) \wedge \exists z\, \varphi_M(U, z, y)$.

Idempotent projection: take $\varphi_{M^E}(U, x, y) \equiv (x = y) \wedge \varphi_M(U, x, y)$.

Product: take $\varphi_{M\cdot N}(U, x, y) \equiv \exists X\, \exists Y\, \exists z\, (U = X \cup Y \wedge \varphi_M(X, x, z) \wedge \varphi_N(Y, z, y))$, i.e., this formula checks the existence of an element z and two "local" tiles, one belonging to M and connecting x to z, and the other belonging to N and connecting z to y. Morover, it checks that all elements of U belong to at least one of the two tiles.

Star: let $\varphi_M(U,x,y)$ be a formula defining a language $M \subseteq T(A)$. We want to define a formula $\varphi_{M^*}(U,x,y)$ whose models are the tiles of M^*. This amounts to saying that there is a (possibly empty) sequence of "local" tiles belonging to M, connecting x to y and "covering" U.

This can be done as follows. Let $R(x_1,x_2)$ be the binary relation defined by

$$R(x_1,x_2) \equiv \exists X\ \varphi_M(X,x_1,x_2)$$

stating that there exists a "local" tile connecting x_1 to x_2, and let R^* denote its reflexive and transitive closure, known to be definable in MSO. The formula $R^*(x_1,x_2)$ checks that there is a (possibly empty) finite connected sequence of "local" tiles of M connecting x_1 to x_2.

Observe that this is not enough to define the expected formula $\varphi_{M^*}(U,x,y)$ since we must also check that all elements of U belong to at least one of these "local" tiles. Equivalently, we have to check that the leftmost element $lf(U)$ and the rightmost element $rg(U)$ of U belong to at least one of the "local" tiles. To this purpose, let $\psi(U,x_1,x_2)$ be the formula defined by

$$\begin{aligned}\psi(U,x_1,x_2) \ \equiv\ & \varphi_M(U,x_1,x_2) \vee \\ & \exists X_1\ \exists X_2\ \exists z_1\ \exists z_2\ (\varphi_M(X_1,x_1,z_1) \wedge R^*(z_1,z_2) \wedge \varphi_M(X_2,z_2,x_2) \\ & \qquad \wedge ((lf(U) \in X_1 \wedge rg(U) \in X_2) \vee (lf(U) \in X_2 \wedge rg(U) \in X_1))).\end{aligned}$$

This formula checks that there is a nonempty sequence of "local" tiles of M connecting x_1 to x_2, such that $lf(U)$ and $rg(U)$ are in the first and last tile, or vice versa. Finally, the formula $\varphi_{M^*}(U,x,y)$ is defined by:

$$\varphi_{M^*}(U,x,y) \ \equiv\ \exists x_1 \exists x_2\ (R^*(x,x_1) \wedge \psi(U,x_1,x_2) \wedge R^*(x_2,y)) \vee\ (U = \{x\} \wedge x = y)$$

where the second part of the disjunction corresponds to the unit tile $(1,1,1) \in M^*$. □

Since finite languages of tiles are clearly MSO-definable, a first consequence of Theorem 4.3 is that regular languages of tiles are MSO-definable. Moreover, since the class REG is not closed under left and right projection (see Lemma 3.7) or since $E(A)$ is not regular (see Lemma 3.8) this inclusion is strict. In other words:

Corollary 4.4. *$REG \subsetneq MSO$.*

A simple characterization of MSO-definable languages of tiles may be obtained via a notion of word congruence induced by a language L of tiles: two words are congruent when in any tile, they can replace each other without altering the membership to L.

Definition 4.5 (Induced word congruence). Let $L \subseteq T(A)$ be a language of tiles. For all $u_0, v_0 \in A^*$, we say that the word u_0 is equivalent to the word v_0 with respect to the language L, which is denoted by $u_0 \simeq_L v_0$, when for all w_1, w_2, w_3 and $w_4 \in A^*$, if $u = (w_1u_0w_2, w_3, w_4)$ and $v = (w_1v_0w_2, w_3, w_4)$, or if $u = (w_1, w_2u_0w_3, w_4)$ and $v = (w_1, w_2v_0w_3, w_4)$, or if $u = (w_1, w_2, w_3u_0w_4)$ and$v = (w_1, w_2, w_3v_0w_4)$, then $u \in L \Leftrightarrow v \in L$ and $u^{-1} \in L \Leftrightarrow v^{-1} \in L$.

By definition, the relation $\simeq_L$ is indeed a congruence in the free monoid A^*. The congruence class of a word $u \in A^*$ is then denoted by $[u]_L$.

Theorem 4.6 (Finite word congruence property). *For every language $L \subseteq T(A)$ of tiles:*

$$\begin{aligned}L = & \bigcup_{(u_1,u_2,u_3) \in L \cap T^+(A)} [u_1]_L \times [u_2]_L \times [u_3]_L \\ & \cup \bigcup_{(u_1,u_2,u_3)^{-1} \in L \cap T^-(A)} ([u_1]_L \times [u_2]_L \times [u_3]_L)^{-1}.\end{aligned}$$

Moreover, L is MSO-definable if and only if $\simeq_L$ is of finite index.

Proof. The expression of L is an immediate consequence of the definition of $\simeq_L$.

Assume that $\simeq_L$ is of finite index. By Myhill-Nerode's theorem, for every $w \in A^*$, the word language $[w]_L \subseteq A^*$ is regular and hence MSO-definable by Büchi-Elgot-Trakhtenbrot's theorem. Then, clearly, for every u_1, u_2 and $u_3 \in A^*$, the three languages $L_1 = [u_1]_L \times \{1\} \times \{1\}$, $L_2 = \{1\} \times [u_2]_L \times \{1\}$ and $L_3 = \{1\} \times \{1\} \times [u_3]_L$ are also MSO-definable; and by the closure properties of *MSO* (Theorem 4.3), the language $L_1 \cdot L_2 \cdot L_3 = [u_1]_L \times [u_2]_L \times [u_3]_L$ and its inverse image are MSO-definable. If $\simeq_L$ is of finite index, then L is a finite union of such MSO-definable languages, thus by Theorem 4.3, the language L itself is MSO-definable.

Conversely, assume that L is MSO-definable. A positive tile $(u, v, w) \in T^+(A)$ can be encoded as the word $c(u, v, w) := u_\ell v_c w_r \in A_\ell^* A_c^* A_r^*$, where A_ℓ, A_c and A_r are three disjoint copies of the alphabet A, and for every word $y \in A^*$, the words y_ℓ, y_c and y_r are the copies of y in these alphabets. It is a straightforward exercise to show that if a tile language $K \subseteq T^+(A)$ is MSO-definable, then so is the language of words $c(K)$. In fact, it is easy to see that this decoding c^{-1} is an MSO-definable transduction, and it is well known that inverse MSO transductions preserve MSO-definability (see Corollary 7.12 of [6]).

Now let $L^+ = L \cap T^+(A)$ and $L^- = L \cap T^-(A)$. Then L^+ and $(L^-)^{-1}$ are languages of positive tiles, and can be encoded as the language of words $M^+ = c(L^+)$ and the language of words $M^- = c((L^-)^{-1})$. Since L is MSO-definable, so are L^+ and $(L^-)^{-1}$, and thus, by the above, their encodings M^+ and M^- are also MSO-definable. By Büchi-Elgot-Trakhtenbrot's theorem both languages M^+ and M^- are regular, and thus their syntactic congruences $\simeq_{M^+}$ and $\simeq_{M^-}$ are of finite index. This implies that $\simeq_L$ is also of finite index. Indeed, for all words $u, v \in A^*$, we have $u \simeq_L v$ if and only if $u_x \simeq_{M^+} v_x$ and $u_x \simeq_{M^-} v_x$ for every $x \in \{\ell, c, r\}$. □

Then, the finite congruence property allows us to prove:

Theorem 4.7 (Simplicity). *A language $L \subseteq T(A)$ of tiles is MSO-definable if and only if L is a finite union of languages of the form $M \times C \times N$ or $(M \times C \times N)^{-1}$, where M, C and N are regular languages of words over A.*

Proof. Let $L \subseteq T(A)$ be a language of tiles. Let $\simeq_L$ be the word congruence associated with L (Definition 4.5). Assume that L is MSO-definable. By Theorem 4.6, we have

$$L = \bigcup_{(u_1,u_2,u_3) \in L \cap T^+(A)} [u_1]_L \times [u_2]_L \times [u_3]_L \\ \cup \bigcup_{(u_1,u_2,u_3)^{-1} \in L \cap T^-(A)} ([u_1]_L \times [u_2]_L \times [u_3]_L)^{-1}$$

and the congruence $\simeq_L$ is of finite index. By Myhill-Nerode's theorem, every $[u_i]_L$ is a regular language of words, which implies that L is a finite union of tile languages of the form $M \times C \times N$ or $(M \times C \times N)^{-1}$, where M, C and N are regular languages of words.

Conversely, assume that L is such a finite union. To prove that L is MSO-definable, it suffices to prove that every tile language of the form $\theta(K) = \{1\} \times K \times \{1\}$ for some regular language K is MSO-definable.

Indeed, for all word languages M, C and $N \subseteq A^*$, we have by Lemma 2.18 that

$$M \times C \times N = \theta(M)^L \cdot \theta(C) \cdot \theta(N)^R$$

and, by Theorem 4.3, the class of MSO-definable languages is closed under left and right projections, finite unions, finite products and inverses.

Now, for every regular language $K \subseteq A^*$, the language $\theta(K)$ is regular (by Theorem 3.5 and because $K \subseteq W(A)$) hence it is MSO-definable (by Corollary 4.4). □

In terms of projection, the previous theorem and its proof arguments ensure that:

Corollary 4.8. *A language of tiles is MSO-definable if and only if it is a finite union of languages of tiles of the form*

$$M_1^L \cdot M_2 \cdot M_3^R \text{ or } M_3^R \cdot M_2^{-1} \cdot M_1^L$$

where M_1, M_2 and M_3 are images by θ of regular languages of words.

As an additional corollary, we retrieve Shepherdson's well-known result [45]:

Corollary 4.9 (Shepherdson's theorem). *Every language of words recognized by a finite-state two-way automaton is regular.*

Proof. Let $\mathcal{A}$ be a finite-state two-way automaton. By definition, the word language recognized by $\mathcal{A}$ is $L_S(\mathcal{A}) = \{u \in A^* : (1, u, 1) \in L_T(\mathcal{A})\}$. Following Remark 3.20, let $\#$ be a new letter not in A. Since 0 is removed from the product of languages of tiles, we have:

$$\theta(\#) \cdot L_T(\mathcal{A}) \cdot \theta(\#) = \theta(\#) \cdot \theta(L_S(\mathcal{A})) \cdot \theta(\#) = \{1\} \times \#L_S(\mathcal{A})\# \times \{1\}$$

By Theorem 3.21, the tile language $L_T(\mathcal{A})$ is regular, thus the language $\theta(\#) \cdot L_T(\mathcal{A}) \cdot \theta(\#)$ is also regular. Hence, by Corollary 4.4, it is MSO-definable and thus, by Theorem 4.7 above, it is of the form $\{1\} \times L \times \{1\}$ for some regular language $L \subseteq \#A^*\#$.

It follows that $L_S(\mathcal{A}) = \{w \in A^* : \#w\# \in L\}$, or, using left and right residual notations (see [43]), $L_S(\mathcal{A}) = \#^{-1}((L)\#^{-1})$. Since regular languages are closed under left or right residuals, we conclude that $L_S(\mathcal{A})$ is a regular language over the alphabet A. □

4.2 *k*-regular languages

We define in this section the notion of k-regular languages of tiles that are defined measuring the nesting depth of the idempotent projection operator in some extended notion of regular expressions.

Definition 4.10 (k-regular languages of tiles). For every $k \in \mathbb{N}$, a tile language $M \subseteq T(A)$ is *k-regular* if either $k = 0$ and M is regular, or $k > 0$ and M can be defined as the result of finitely many additions, multiplications and star operations over both $(k-1)$-regular tile languages and idempotent projections of $(k-1)$-regular tile languages. The class of k-regular tile languages is denoted by *k-REG*.

Example 4.11. The language $E(A)$ of non-zero idempotent tiles is 1-regular since $T(A)$ is regular (see Example 3.3) and $E(A) = (T(A))^E$.

The Robustness Theorem 4.3 and the Simplicity Theorem 4.7 yield a characterization of k-regular tile languages.

Theorem 4.12. *For every $k > 0$, we have k-REG = MSO = 1-REG.*

Proof. By construction *REG* $\subseteq$ *1-REG* $\subseteq$ *2-REG* $\subseteq \cdots \subseteq$ *k-REG* $\subseteq \cdots$ and by Corollary 4.4 and the Robustness Theorem 4.3 we have *k-REG* $\subseteq$ *MSO* for every $k \in \mathbb{N}$.

But we observe that for every regular language of words M, we have that $\theta(M) \in$ *REG* and

$$\theta(M)^L = (\theta(A^*)^{-1} \cdot \theta(M))^E \text{ and } \theta(M)^R = (\theta(M) \cdot \theta(A^*)^{-1})^E .$$

By definition, this means that both $\theta(M)^L$ and $\theta(M)^R$ are *1-REG* hence, by Corollary 4.8 and Lemma 3.4, every MSO-definable languages of tiles is in *1-REG*. It follows that *MSO* $\subseteq$ *1-REG*. □

Theorem 4.12 ensures that *1-REG* = *k-REG* for every $k \geq 1$. This suggests that there is no point in defining and studying k-regular tile languages by themselves. However, the tight correspondence obtained in the next section (Theorem 4.18) between k-regular expressions and k-pebble automata suggests that, in a way analogous to Globerman and Harel's result [17], a *succinctness* hierarchy is induced by the number of allowed pebbles. More precisely:

Conjecture 4.13. *k-REG* expressions are k-fold exponentially more succinct than *1-REG* expressions.

4.3 Two-way pebble automata

In this section, we define the notion of k-pebble two-way automata on tiles. Then, we show that k-pebble automata capture k-regular tile languages. Informally, a pebble automaton is a two-way automaton that has the capacity, from time to time, to drop and lift pebbles placed *between* letters of the input word.

Here we consider *invisible* pebbles in the sense of [11]: at any time, only the last pebble dropped may be seen by the automaton, and only by lifting this pebble. Also, as we will consider automata with a bounded number of pebbles, the pebbles we use are unmarked.

The k-(invisible, unmarked)-pebble automata are particular cases of the k-(visible, marked)-pebble automata of [17, 12], whose transitions may be governed by the presence or absence of pebbles on the current position. The more general case of infinitely many invisible (marked) pebbles is considered in [26] when studying walking automata on birooted trees or graphs.

Our proposed definition is inspired by stack of stack automata as studied in [15]. Indeed, each pebble dropped (resp. lifted) during a run can be modeled as pushing (resp. popping) a new stack on (resp. the top stack from) the main stack, kept in the run configuration. In this case, as opposed to the case of trees or graphs [26], these secondary stacks are rather simple. They are integers since it suffices to remember at every step the distance between the reading head and the last pebble that has been dropped (or the starting position).

Definition 4.14 (Pebble automata). A *finite-state pebble two-way automaton* over an alphabet A is a quadruple $\mathcal{A} = \langle Q, I, F, \Delta \rangle$ with a finite set of states Q, a set of initial states $I \subseteq Q$, a set of final states $F \subseteq Q$, and a transition table $\Delta : (A \cup \bar{A} \cup \{1^+, 1^-\}) \to \mathcal{P}(Q \times Q)$.

Informally, for every $a \in A$, the set of transitions $\Delta(a)$ tells how the letter a can be read from left to right, and the set of transitions $\Delta(\bar{a})$ tells how that letter can be read from right to left. The set of transitions $\Delta(1^+)$ tells how a pebble can be dropped, and the set of transitions $\Delta(1^-)$ tells how a pebble can be lifted.

Runs are then defined via *position configurations*: non-empty finite sequences of (positive or negative) integers $p = z_0 z_1 \cdots z_k$ for some integer $k \geq 0$ and $z_i \in \mathbb{Z}$ for every $0 \leq i \leq k$. The intended meaning of such a position configuration p is that k pebbles are dropped on the input tile, and z_i is the distance between the positions of the ith dropped pebble and the $(i+1)$th dropped pebble, with the initial position of the reading head modeled as a sort of 0th pebble and its current position as a sort of $(k+1)$th pebble. This intention is made more precise in the next two definitions and in the remark afterward.

Definition 4.15 (Runs of pebble automata). A *run* of the pebble automaton $\mathcal{A}$ is a finite sequence

$$\rho = (q_0, p_0)a_1(q_1, p_1) \cdots (q_{n-1}, p_{n-1})a_n(q_n, p_n) \tag{1}$$

where $n \geq 0$, $q_0, \ldots, q_n \in Q$, $p_0, \ldots, p_n \in \mathbb{Z}^+$ and $a_1, \ldots, a_n \in A \cup \bar{A} \cup \{1\}$, such that $a_1 \cdots a_n \in W(A)$ and for every $1 \leq i \leq n$ one of the following conditions is satisfied:

(1) $(q_{i-1}, q_i) \in \Delta(a_i)$, $a_i \neq 1$, $p_{i-1} = pz$ for some $p \in \mathbb{Z}^*$ and $z \in \mathbb{Z}$, and $p_i = p(z + \delta(a_i))$ with $\delta(a_i) = 1$ if $a_i \in A$ and $\delta(a_i) = -1$ if $a_i \in \bar{A}$.
(2) $(q_{i-1}, q_i) \in \Delta(1^+)$, $a_i = 1$ and $p_i = p_{i-1}0$.
(3) $(q_{i-1}, q_i) \in \Delta(1^-)$, $a_i = 1$ and $p_{i-1} = p_i 0$.

Then we put:

Definition 4.16 (k-pebble recognizability). A run ρ of $\mathcal{A}$, as in Equation (1), *uses at most k pebbles* when, for every $0 \leq i \leq n$, the length of the sequence p_i is at most $k+1$.

For $k \in \mathbb{N}$, a *tile* u is *k-recognized* by $\mathcal{A}$ when there exists a run ρ of $\mathcal{A}$, as in Equation (1), that uses at most k pebbles, such that $u = \theta(a_1 \cdots a_n)$, with $(q_0, p_0) \in I \times \{0\}$ and $(q_n, p_n) \in F \times \mathbb{Z}$.

The *tile language* $L \subseteq T(A)$ is *k-recognized* by $\mathcal{A}$ if L is the set of tiles that are k-recognized by $\mathcal{A}$.

Let *k-P2WA* denote the class of tile languages k-recognized by finite-state pebble two-way automata.

Remark 4.17. A simple check shows that, in the definition above, if $u = (u_1, u_2, u_3)$, then $p_n = |u_2|$ when $u_2 \in A^*$ and $p_n = -|u_2|$ when $u_2 \in \bar{A}^*$, i.e., p_n is the relative position of the output root of u compared to the input root of u.

More generally, when interpreting a run as a back and forth traversal of the tile $u = \theta(a_1 a_2 \cdots a_n)$ from its input root to its output root (as in Remark 3.19), one can show that if $p_0 = 0$ then every other position configuration p_i is of the form $p_i = z_0 z_1 \cdots z_m$ with the integers z_0, z_1, …, z_m interpreted as follows. The integer z_0 gives the relative position of the first dropped pebble (or the reading head when $m = 0$) compared to the start of the run (i.e., the input root of u). For every $0 < j < m$, the integer z_j gives the relative position of the $(j+1)$th pebble compared to the position of the jth pebble. Last, the integer z_m gives the relative position of the reading head compared to the position of the $(m-1)$th pebble (or its initial position when $m = 0$). The pebble automaton $\mathcal{A}$ keeps track of such relative positions by counting the "number" of letters it reads in $a_1 \cdots a_n$, counted positive when in A (i.e., when moving to the right on $\theta(a_1 \cdots a_n)$) and negative when in $\bar{A}$ (i.e., when moving to the left).

Indeed, dropping a pebble amounts to pushing 0, the new position of the reading head compared to that pebble. Reading $a \in A \cup \bar{A}$, the position of the head compared to the last dropped pebble (which is on top of the stack) is changed by $\delta(a)$. Lifting a pebble amounts to popping the position of the head compared to the position of that pebble. That position must be 0; that ensures that the head has moved back to the position it had when the pebble was dropped.

Theorem 4.18. *k-REG = k-P2WA for every $k \in \mathbb{N}$.*

Proof. Follows from Lemmas 4.19 and 4.20 below. □

Lemma 4.19. *Every language of tiles k-recognized by a finite-state pebble two-way automaton is k-regular.*

Proof. Let $\mathcal{A} = \langle Q, I, F, \Delta \rangle$ be a finite-state pebble two-way automaton. For every pair of states $(p, q) \in Q \times Q$ and every integer $k \geq 0$, let $T^k_{p,q} \subseteq T(A)$ denote the language of tiles k-recognized by the automaton $\langle Q, \{p\}, \{q\}, \Delta \rangle$. Let $C^k_{p,q}$ denote the associated set of idempotent tiles $C^k_{p,q} = (T^k_{p,q})^E$.

It is easy to show that the languages $T^k_{p,q}$ form the least solution of the set of equations defined, for every $p, q \in Q$ and every $k \geq 0$, by:

$$T^k_{p,q} = \delta_{p,q} + \sum_{a \in A \cup \bar{A}} \sum_{(p,r) \in \Delta(a)} \theta(a) \cdot T^k_{r,q} + \sum_{(p,p') \in \Delta(1^+)} \sum_{(r',r) \in \Delta(1^-)} C^{k-1}_{p',r'} \cdot T^k_{r,q}$$

with $C^{-1}_{p',r'} = \emptyset$ and $\delta_{p,q} = \{1\}$ when $p = q$ and $\emptyset$ otherwise.

Indeed, we just mimic in these equations all the possible cases to build a run. Either some letter $a \in A \cup \bar{A}$ is read, or a pebble is used. Observe that $T^k_{p,q}$ only depends on tile languages of the form $T^{k'}_{p',q'}$ with $k' \leq k$ or of the form $C^{k'}_{p',q'} = (T^{k'}_{p',q'})^E$ with $k' < k$. Thus no circular dependency involves idempotent projections. It follows that this system can be solved by induction on $k \in \mathbb{N}$, using Gaussian elimination, by Lemma 3.1. The tile language of tiles k-recognized by $\mathcal{A}$ is $\sum_{(p,q)\in I\times F} T^k_{p,q}$, and thus k-regular. □

Lemma 4.20. *Every k-regular language of tiles is k-recognized by a finite-state pebble two-way automaton.*

Proof. For $k = 0$ the result follows from Theorem 3.21. For $k \geq 1$ the proof is by induction on the syntactic complexity of k-regular expressions, combining pebble automata. We start with the construction for the idempotent projection.

Given a pebble automaton $\mathcal{A} = \langle Q, I, F, \Delta\rangle$ and its k-recognized tile language L, we define the automaton $\mathcal{A}' = \langle Q', I', F', \Delta'\rangle$ by $Q' = Q \cup \{q_0, q_f\}$ with q_0 and q_f two new states, $I' = \{q_0\}$, $F' = \{q_f\}$, and, for every $a \in A \cup \bar{A}$, $\Delta'(a) = \Delta(a)$, $\Delta'(1^+) = \Delta(1^+) \cup (\{q_0\} \times I)$ and $\Delta'(1^-) = \Delta(1^-) \cup (F \times \{q_f\})$. It is straightforward to check that the tile language L^E is $(k+1)$-recognized by the automaton $\mathcal{A}'$.

Now let $\mathcal{A}_1 = \langle Q_1, I_1, F_1, \Delta_1\rangle$ and $\mathcal{A}_2 = \langle Q_2, I_2, F_2, \Delta_2\rangle$ be two pebble automata,k-recognizing the tile languages L and M. We assume that Q_1 and Q_2 are disjoint. Clearly, $L+M$ is k-recognized by the automaton $\langle Q_1 \cup Q_2, I_1 \cup I_2, F_1 \cup F_2, \Delta\rangle$ where $\Delta(a) = \Delta_1(a) \cup \Delta_2(a)$ for every $a \in A \cup \bar{A}$.

Since, in general, $(L + \{1\}) \cdot (M + \{1\}) = L \cdot M + L + M + \{1\}$ and $(L + \{1\})^* = L^*$, we may assume in the remaining cases that $I_1 = \{q_{0,1}\}$ and $F_1 = \{q_{f,1}\}$ with $q_{0,1} \neq q_{f,1}$, and similarly for $\mathcal{A}_2$. Moreover, we may assume that if $(q, q') \in \Delta_1(a)$ for $a \in A \cup \bar{A}$, then $q \neq q_{f,1}$ and $q' \neq q_{0,1}$, and similarly for $\mathcal{A}_2$. Finally, since the number of pebbles used is finite, we may assume that $\mathcal{A}_1$ keeps track of the number of dropped pebbles in its finite state, and so,if $(q_{0,1}, 0) \cdots (q_{f,1}, p)$ is a run of $\mathcal{A}_1$, then $p \in \mathbb{Z}$, and similarly for $\mathcal{A}_2$.

Under these assumptions, a pebble automaton for L^* is obtained from $\mathcal{A}_1$ by identifying the states $q_{0,1}$ and $q_{f,1}$. Moreover, a pebble automaton for $L \cdot M$ is obtained from the automaton $\langle Q_1 \cup Q_2, \{q_{0,1}\}, \{q_{f,2}\}, \Delta\rangle$, where Δ is defined as above, by identifying the states $q_{f,1}$ and $q_{0,2}$. □

It is well known that the language of words k-recognized by a finite-state pebble two-way automaton is regular [12, 13, 11]. The next corollary states this for our pebble automata. We define the language of words k-recognized by pebble automaton $\mathcal{A}$ to consist of all words $u \in A^*$ such that the tile $(1, u, 1)$ is k-recognized by $\mathcal{A}$.

Corollary 4.21. *Every language of words k-recognized by a finite-state pebble two-way automaton is regular.*

Proof. By Lemma 4.19 and Theorem 4.12 the tile language k-recognized by a finite-state pebble two-way automaton is MSO-definable. The remainder of the proof is the same as in the proof of Corollary 4.9. □

5 Conclusion

Studying languages of overlapping tiles, equivalently subsets of McAlister monoids, we have considered several classes of languages: recognizable languages, regular languages, k-regular languages and MSO-definable languages, obtaining a strict though finite hierarchy

$$\mathit{REC} \subsetneq \mathit{REG} = \mathit{0\text{-}REG} \subsetneq \mathit{1\text{-}REG} = \mathit{k\text{-}REG} = \mathit{MSO}$$

for every $k \geq 1$, with a clear connection between k-regular expressions and languages k-recognized by pebble automata.

Concerning further works, we have already mentioned the question of the succinctness hierarchy possibly induced by the maximum number of pebbles or, equivalently, the nesting depth of idempotent projections. The limit case of runs of pebble automata with no bound on the number of allowed pebbles could also be studied.

The tight connection already provided between (invisible) pebbles in two-way pebble automata and idempotent projection within inverse semigroup theory reenforces the idea that the latter, a robust mathematical theory, can be used to strengthen the former, a somehow quite adhoc theory arising from application perspectives.

An intriguing related class of languages of tiles is the class $BOOL(REG)$ of finite boolean combinations of regular languages. It is obviously included in the class of MSO-definable languages, but it is by no means clear whether the inclusion is strict. Another further work would be to relate the hierarchy with classes of algebraically recognizable languages of tiles, as defined in [20, 24, 25].

Pebble automata can be seen as a restricted class of pushdown automata. This suggests that context-free grammars over languages of tiles could also be studied. The presence of overlaps together with compatibility constraints in the tile product clearly shows that the underlying word languages would no longer be context-free. Still, studying these languages could lead to alternative characterizations of weaker classes of context-sensitive languages such as, for instance, the class of mildly context-sensitive grammars [49] much studied in computational linguistics.

Last, the inverse monoid approach proposed here for studying the behavior of two-way automata could perhaps be extended to two-way transducers.

Acknowledgement

The authors wish to express the deepest gratitude to anonymous referees for the incredibly high quality, depth and length of their reports on former versions of the present paper. Thanks to their comments some entire sections and proof arguments have been entirely made anew. Of course, only the authors should be blamed for remaining mistakes or clumsiness.

References

1. F. S. Almeida. *Algebraic Aspects of Tiling Semigroups.* PhD Thesis, Universidade de Lisboa, Faculdade de Ciências Departamento de Matemática, Lisboa, Portugal, 2010.
2. M. Anselmo. Automates et code zigzag. *ITA*, 25:49–66, 1991.
3. J. Berstel. *Transductions and Context-Free Languages.* Teubner Verlag, 1979.
4. F. Berthaut, D. Janin, and B. Martin. Advanced synchronization of audio or symbolic musical patterns: an algebraic approach. *International Journal of Semantic Computing*, 6(4):409–427, 12 2012.
5. J.-C. Birget. Concatenation of inputs in a two-way automaton. *Theor. Comp. Sci.*, 63(2):141 – 156, 1989.
6. B. Courcelle and J. Engelfriet. *Graph structure and monadic second-order logic, a language theoretic approach*, volume 138 of *Encyclopedia of mathematics and its applications.* Cambridge University Press, 2012.
7. A. Dicky and D. Janin. Embedding finite and infinite words into overlapping tiles. In *Developments in Language Theory (DLT)*, volume 8633 of *LNCS*, pages 339–347. Springer, 10 2014.
8. Do Long Van, B. LeSaëc, and I. Litovsky. On coding morphisms for zigzag codes. *ITA*, 26:565–580, 1992.

9. E. Dubourg and D. Janin. Algebraic tools for the overlapping tile product. In *Language and Automata Theory and Applications (LATA)*, volume 8370 of *LNCS*, pages 335 – 346. Springer, 03 2014.
10. J. Engelfriet, H. J. Hoogeboom, and J.-P. Van Best. Trips on trees. *Acta Cybern.*, 14(1):51–64, 1999.
11. J. Engelfriet, H. J. Hoogeboom, and B. Samwel. XML transformation by tree-walking transducers with invisible pebbles. In *Principles of Database System (PODS)*. ACM, 2007.
12. J. Engelfriet and H.J. Hoogeboom. Tree-walking pebble automata. In J. Karhumäki, H. Maurer, G. Paun, and G. Rozenberg, editors, *Jewels are forever, contributions to Theoretical Computer Science in honor of Arto Salomaa*, pages 72–83. Springer, 1999.
13. J. Engelfriet and H.J. Hoogeboom. Automata with nested pebbles capture first-order logic with transitive closure. *Logical Methods in Computer Science*, 3, 2007.
14. J. Fountain, G. Gomes, and V. Gould. The free ample monoid. *Int. Jour. of Algebra and Comp.*, 19:527–554, 2009.
15. S. Fratani and G. Sénizergues. Iterated pushdown automata and sequences of rational numbers. *Ann. Pure Appl. Logic*, 141(3):363–411, 2006.
16. V. Geffert and L. Istonová. Translation from classical two-way automata to pebble two-way automata. *RAIRO - Theor. Inf. and Applic.*, 44(4):507–523, 2010.
17. N. Globerman and D. Harel. Complexity results for two-way and multi-pebble automata and their logics. *Theor. Comp. Sci.*, 169(2):161–184, 1996.
18. P. Hudak and D. Janin. Tiled polymorphic temporal media. In *Work. on Functional Art, Music, Modeling and Design (FARM)*, pages 49–60. ACM Press, 2014.
19. D. Janin. Quasi-inverse monoids (and premorphisms). Research report RR-1459-12, LaBRI, Université de Bordeaux, 04 2012.
20. D. Janin. Quasi-recognizable vs MSO definable languages of one-dimensional overlapping tiles. In *Mathematical Found. of Comp. Science (MFCS)*, volume 7464 of *LNCS*, pages 516–528, 09 2012.
21. D. Janin. Vers une modélisation combinatoire des structures rythmiques simples de la musique. *Revue Francophone d'Informatique Musicale (RFIM)*, 2, 09 2012.
22. D. Janin. Algebras, automata and logic for languages of labeled birooted trees. In *Int. Col. on Aut., Lang. and Programming (ICALP)*, volume 7966 of *LNCS*, pages 318–329. Springer, 07 2013.
23. D. Janin. On languages of one-dimensional overlapping tiles. In *Int. Conf. on Current Trends in Theo. and Prac. of Comp. Science (SOFSEM)*, volume 7741 of *LNCS*, pages 244–256. Springer, 01 2013.
24. D. Janin. Overlaping tile automata. In *8th Int. Computer Science Symp. in Russia (CSR)*, volume 7913 of *LNCS*, pages 431–443. Springer, 06 2013.
25. D. Janin. On languages of labeled birooted trees: Algebras, automata and logic. *Information and Computation*, 2014.
26. D. Janin. Walking automata in the free inverse monoid. Research report, LaBRI, Université de Bordeaux, 2015.
27. D. Janin, F. Berthaut, and M. Desainte-Catherine. Multi-scale design of interactive music systems : the libTuiles experiment. In *Sound and Music Comp. (SMC)*, 2013.
28. D. Janin, F. Berthaut, M. DeSainte-Catherine, Y. Orlarey, and S. Salvati. The T-calculus : towards a structured programming of (musical) time and space. In *Work. on Functional Art, Music, Modeling and Design (FARM)*, pages 23–34. ACM Press, 2013.
29. J. Kellendonk. The local structure of tilings and their integer group of coinvariants. *Comm. Math. Phys.*, 187:115–157, 1997.
30. J. Kellendonk and M. V. Lawson. Tiling semigroups. *Journal of Algebra*, 224(1):140 – 150, 2000.
31. J. Kellendonk and M. V. Lawson. Universal groups for point-sets and tilings. *Journal of Algebra*, 276:462–492, 2004.
32. M. Kunc and A. Okhotin. Describing periodicity in two-way deterministic finite automata using transformation semigroups. In *Developments in Language Theory (DLT)*, volume 6795 of *LNCS*, pages 324–336. Springer, 2011.
33. M. V. Lawson. *Inverse Semigroups : The theory of partial symmetries*. World Scientific, 1998.
34. M. V. Lawson. McAlister semigroups. *Journal of Algebra*, 202(1):276 – 294, 1998.

35. B. LeSaëc, I. Litovsky, and B. Patrou. A more efficient notion of zigzag stability. *ITA*, 30(3):181–194, 1996.
36. S. W. Margolis and J.-E. Pin. Languages and inverse semigroups. In *Int. Col. on Aut., Lang. and Programming (ICALP)*, volume 172 of *LNCS*, pages 337–346. Springer, 1984.
37. D.B. McAlister. Inverse semigroups which are separated over a subsemigroups. *Trans. Amer. Math. Soc.*, 182:85–117, 1973.
38. W. D. Munn. Free inverse semigroups. *Proceeedings of the London Mathematical Society*, 29(3):385–404, 1974.
39. J.-P. Pécuchet. Automates boustrophedon, semi-groupe de Birget et monoide inversif libre. *ITA*, 19(1):71–100, 1985.
40. M. Petrich. *Inverse semigroups*. Wiley, 1984.
41. J-.E. Pin. Chap. 10. Syntactic semigroups. In G. Rozenberg and A. Salomaa, editors, *Handbook of formal languages, Vol. I*, pages 679–746. Springer-Verlag, 1997.
42. M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114 –125, april 1959.
43. J. Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.
44. H. E. Scheiblich. Free inverse semigroups. *Semigroup Forum*, 4:351–359, 1972.
45. J. C. Shepherdson. The reduction of two-way automata to one-way automata. *IBM J. Res. Dev.*, 3:198–200, April 1959.
46. P. V. Silva. On free inverse monoid languages. *ITA*, 30(4):349–378, 1996.
47. W. Thomas. Chap. 7. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages, Vol. III*, pages 389–455. Springer-Verlag, Berlin Heidelberg, 1997.
48. M. Y. Vardi. A note on the reduction of two-way automata to one-way automata. *Information Processing Letters*, 30:261–264, 1989.
49. D.J. Weir. *Characterizing mildly context-sensitive grammar formalisms*. PhD thesis, University of Pennsylvania, 1988.